THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Toulouse III Paul Sabatier (UT3 Paul Sabatier)

Discipline ou spécialité :

Informatique

Présentée et soutenue par :
Damien DJAOUTI

le : lundi 28 novembre 2011

Titre :
Serious Game Design
Considérations théoriques et techniques sur la création de jeux vidéo à vocation utilitaire

---

Ecole doctorale :
Mathématiques Informatique Télécommunications (MITT)

Unité de recherche :
Institut de Recherche en Informatique de Toulouse (IRIT)

Directeur(s) de Thèse :

Pr. Jean-Pierre JESSEL

Rapporteurs :

Pr. Pascal ESTRAILLIER
Pr. Franck TARPIN-BERNARD

Membre(s) du jury :

Mr Gilles GALLEE
Mr Abdelkader GOUAICH
Mme Josette HOSPITAL
Pr. Philippe PALANQUE
Pr. Jean-Christophe ROUTIER

REMERCIEMENTS

Je tiens tout d'abord à remercier ***Jean-Pierre Jessel***, mon directeur de thèse, pour son encadrement et ses nombreux conseils avisés. Je tiens également à remercier ***Gilles Methel*** pour son encadrement et ses sages conseils lors du démarrage de ce travail. J'ai également un profond sentiment de gratitude pour ***Julian Alvarez***, avec qui nous avons menés, et continuons à mener, de nombreux travaux de recherche dans le passionnant univers des Serious Games. Merci à tous les trois pour m'avoir permis de découvrir le secteur de la recherche. Toujours dans le monde académique, je remercie également pour leurs conseils avisés, leur idées passionnantes ou leurs remarques pertinentes : ***Olivier Rampnoux***, ***Michel Lavigne***, ***Pierre Molinier***, ***Patrick Mpondo-Dicka, Yves Duthen, Cédric Sanza, Stéphane Sanchez, Patrice Torguet, Véronique Gaildrat, Julien Honnorat, François Donato…***
Je remercie aussi ***Pascal Estraillier*** et ***Franck Tarpin-Bernard*** pour avoir accepté d'être les rapporteurs de cette thèse, ainsi que pour leurs remarques constructives et pertinentes. Je remercie également les membres du jury qui ont accepté d'évaluer ce travail : ***Abdelkader Gouaich***, ***Gilles Gallée***, ***Josette Hospital***, ***Philippe Palanque*** et ***Jean-Christophe Routier***.

Le travail présenté dans cette thèse n'aurait pu avoir lieu sans la convention CIFRE qui l'accompagne. Je tiens donc à remercier très chaleureusement ***Josette Hospital***, principale architecte du complexe assemblage de ce partenariat « université – collectivité territoriale – entreprise privée ». Je remercie également ***Jean-Pierre Jessel***, ***Alain Garès***, ***Bernard Keller***, ***Joseph Carles***, ***Françoise Laborde***, ***Eric Poisson***, ***Franck Perret*** et ***Julian Alvarez*** pour avoir permis à cette convention CIFRE d'exister. Je souhaite également remercier tous mes collègues du volet « industriel » de mon travail de recherche. Du coté de la ***ludothèque Odyssud***, je remercie ***Josette Hospital, Noémie Birien***, ***Brigitte Giovannoni***, ***Jeanne Servat, Rosalia***, ***Mélissa***, ***Charlène***, ***Pierrine, Laurent*** et tous mes autres collègues pour leur accueil chaleureux et les nombreuses expériences partagées. A la ***SEM Constellation***, je remercie profondément ***Florence Willm*** et ***Pauline Gambart*** pour leur très grande implication dans les Serious Games dédiés à l'écoquartier *Andromède*. Chez ***OKTAL***, je remercie ***Eric Poisson***, ***Franck Perret***, ***Gilles Gallée, Anton Minko*** et ***Viviane Pena*** pour m'avoir fait découvrir le vaste univers de la simulation professionnelle.

Je remercie aussi toutes les personnes que j'oublie de citer mais que j'ai eu l'occasion de croiser durant ces dernières années. Par leurs idées, leurs remarques, leurs travaux ou tout simplement par les bons moments que nous avons partagés, elles ont directement contribué à rendre d'autant plus passionnante l'aventure humaine que représente une thèse.

Enfin, je remercie ma famille et mes amis pour leurs encouragements et leur soutien continu durant toutes ces années de recherche. Merci notamment à toi ***Elodie***, pour ton soutien et ton amour durant la rédaction de ce mémoire. Merci à toi aussi ***Xavier***, pour ces nombreuses heures que nous avons passé, il y a déjà quelques années, à créer des jeux vidéo amateurs grâce à *M.U.G.E.N.*, *The Games Factory* et *Multimédia Fusion 1.5*. Mais surtout, je tiens à remercier infiniment mes parents, ***Evelyne*** et ***M'hammed*** pour leur support, leur soutien, leurs encouragements et leur affection continue durant mes très nombreuses années d'études supérieures, qui aboutissent à la rédaction de cette thèse. Sans vous, ce travail n'aurait jamais vu le jour. Je tiens donc à vous dédier cette thèse, ainsi que tout le travail qu'elle représente…

ABSTRACT

Nowadays, videogames are an important part of popular culture. Although videogames are most famous as leisure, the wave of « Serious Games » aims to use them for a wide range of serious purposes: *teaching, communication, therapy, professional training, advertising, political propaganda…*

This thesis focuses on the design of such « Serious Games ». More specifically, we will try to identify means able to ease the creation of Serious Games. To achieve this goal, we will first concentrate on the analysis of means used to ease the creation of entertainment videogames. We will then try to evaluate how these means can be relevant to ease the creation of Serious Games. Through this process, we will specify the difference between the design of an entertainment videogame and a Serious Game. This research was conducted with the help of an industrial partnership involving three companies and institutions. The main contributions of this thesis are:

- A global view of the « Serious Game » field through a corpus of 2218 titles.

- An analysis of the « theoretical » tools able to ease the creation of videogames, backed with four case studies coming from our industrial partners. This study of the « theoretical » side of the Serious Game Design process will result in a synthetic methodology to create Serious Games.

- An analysis of the « technical » tools able to ease the creation of videogames, based on a corpus of 400 software tools. This study will result in an open and collaborative database of videogame creation tools.

- An evaluation of these « theoretical » and « technical » tools through a university course whose goal is to teach the basics of Game Design. This evaluation will point out several limitations in the available tools. This study will thus result in a specification of desired features for a tool able to ease the creation of Serious Games in an educational environment.

- Two software experimentations aiming to create a tool embedding all the features listed in the previous specification. The first experiment failed to reach this goal, but the second one shows a promising mid-term result: this prototype tool was successfully used to create one of the Serious Games designed for our industrial partners.

Keywords: *Serious Game, Game Design, videogames, creation, easing, pedagogy, constructionnism, authorware, game creation tools, modding.*

RESUME

Les jeux vidéo font aujourd'hui partie du paysage culturel de notre société. Si leur terrain de prédilection reste le domaine des loisirs, le courant du « Serious Game » permet d'utiliser les jeux vidéo pour de nombreuses finalités « sérieuses » : *apprentissage scolaire, communication, thérapie, formation professionnelle, publicité, militantisme…*

Le travail présenté dans cette thèse porte sur la création de ces « jeux vidéo à vocation utilitaire ». Plus précisément, nous nous interrogeons sur les approches permettant de faciliter la création de Serious Games. Pour cela, nous recensons et analysons les approches existantes de facilitation de la création de jeux vidéo de divertissement. Nous évaluons ensuite la pertinence de ces approches pour faciliter la création de Serious Games, tout en cherchant à caractériser les éventuelles spécificités de leur processus créatif. Réalisé dans le cadre d'une convention *CIFRE*, notre travail de recherche propose les contributions suivantes :

- Une vue d'ensemble des « Serious Games » à travers un corpus de 2218 titres.

- Une analyse des outils « théoriques » facilitant la création vidéoludique, appuyée par quatre retours d'expérience *CIFRE* sur la réalisation de Serious Games. Cette étude des considérations « théoriques » nous permet de proposer une méthodologie de conception synthétique adaptée aux Serious Games.

- Une analyse des outils « techniques » facilitant la création vidéoludique, basée sur l'étude d'un corpus de 400 logiciels. Cette étude des considérations « techniques » nous permet de proposer une base de données collaborative en ligne simplifiant la recherche d'outils techniques grâce à un système de classification adapté.

- L'évaluation de ces approches théoriques et techniques de facilitation de la création de Serious Games dans un contexte d'application précis : des cours universitaires reposant sur la création vidéoludique. Cette évaluation des approches identifiées nous permet de proposer un « cahier des charges » définissant les caractéristiques d'un « outil idéal » pour faciliter la création de Serious Games en contexte pédagogique.

- Deux expérimentations logicielles visant à réaliser un nouvel outil répondant à ce « cahier des charges ». Si la première expérimentation n'a pas abouti, la seconde propose des résultats intermédiaires encourageants : cette ébauche d'outil a permis de créer un des Serious Games du *CIFRE* accompagnant cette thèse.

Mots-clés : *Serious Game, Game Design, jeux vidéo, utilitaire, conception, création, facilitation, pédagogie, constructionnisme, logiciel auteur, usines à jeux, modding.*

TABLE DES MATIERES

[Partie II]
Considérations théoriques sur le Serious Game Design

[Partie III]
Considérations Techniques sur le Serious Game Design

[Partie IV]
Une application pédagogique du Serious Game Design

1 INDEX DES FIGURES

2 INDEX DES TABLEAUX

INTRODUCTION

1 PREAMBULE

Né dans les années 1980, je fais partie de la génération des *« digital natives »* (Prensky, 2007). J'ai donc grandi avec le jeu vidéo en tant que loisir. Personnellement, ce loisir ne m'a pas seulement touché en tant que joueur, mais également en tant que créateur. Dès l'âge de 13 ans, j'ai éprouvé le désir de créer mon propre jeu vidéo. Seulement, la création d'un jeu vidéo est un véritable métier qui requiert un haut niveau de compétences *(p.46)*. Alors que ce rêve me semblait inaccessible, j'ai découvert que le jeu *Duke Nukem 3D* ***(3D Realms, 1996)*** était livré avec un logiciel de type *« éditeur de niveau » (p.50)*. Dépourvu de compétence technique particulière dans le domaine, j'ai pu grâce à cet outil créer mes propres « niveaux » pour ce titre. En utilisant les autres outils livrés avec ce même jeu, j'ai également pu en modifier les règles et les graphismes, créant ainsi un *« mod » (p.51)*. Aussi formidable que fut cette possibilité de créer du contenu pour des jeux existants, l'adolescent que j'étais regrettait cependant d'être limité à de simples « modifications » : j'éprouvais toujours l'envie de créer « mon propre jeu vidéo ». Je trouvais enfin un logiciel répondant à mes attentes avec *Klik & Play (p.178)*, un outil de type *« usine à jeux » (p.49)*. À l'aide de cet outil je pouvais enfin créer mes propres jeux vidéo sans pour autant posséder l'immense savoir des professionnels de l'industrie. Alors certes, les jeux que j'étais ainsi en mesure de créer étaient des plus simples : à base de graphismes en 2D, leurs principes de jeu allaient du *« casse-briques »* au petit *« jeu de plateforme »*. Mais peu importait le fait que mes créations n'étaient pas technologiquement comparables avec celles des professionnels. L'essentiel était que cette *« usine à jeux »* me permettait de créer mes propres jeux vidéo, malgré mon absence totale de compétences dans le domaine. J'ai donc continué à créer des jeux vidéo amateurs grâce à plusieurs logiciels du même type : *M.U.G.E.N. (p.172)*, *The Games Factory (p.178)*, *3D Rad (p.184)* et *Multimedia Fusion (p.178)*. Cette passion d'adolescent a même influencée le choix de mon premier emploi, à savoir développeur-concepteur multimédia. J'ai alors pu travailler professionnellement sur des projets de jeux vidéo, tels que les jeux éducatifs *Charivari de Chat-Malô : Pacha* ***(Magelis, 2003)***, *La Cité Romaine* ***(France 5, 2005)*** et *Mobliclic* ***(Milan Presse, 1998-2011)***, et sur des jeux publicitaires pour le compte de la société ***JA-Games***.

2 PROBLEMATIQUE DE RECHERCHE

Ces exemples de jeux vidéo destinés à l'éducation ou à la publicité font partie d'un courant que l'on appelle aujourd'hui le *« Serious Game » (p.18)*. Regroupant des *« jeux dont la finalité première est autre que le simple divertissement »* (Michael & Sande Chen, 2005), ce courant permet au jeu vidéo de toucher de nouveaux secteurs *(p.32)* : *éducation, santé, publicité, formation professionnelle, militantisme, art numérique…* Le fait que le jeu vidéo traite ainsi de nombreuses thématiques « sérieuses » n'est pas sans amener plusieurs problématiques concernant la création de ces « Serious Games ». Traditionnellement, la conception des titres produits par l'industrie du jeu vidéo de divertissement s'appuie sur les compétences d'un *« Game Designer »*, dont le savoir professionnel lui permet de concevoir un jeu divertissant pour le joueur. Mais avec les Serious Games, la phase de conception repose rarement sur les seules connaissances du *« Game Designer »*. S'il s'agit par exemple de réaliser un jeu vidéo traitant de la gestion de certaines maladies chroniques, il faudra impliquer des médecins spécialistes de ces maladies dans la création du Serious Game. De même, lors de la création d'un jeu destiné à former des employés au processus de certification d'une norme particulière, la présence d'un expert de cette norme sera indispensable pour

assurer la pertinence du titre. Et que dire alors des nombreux Serious Games utilisés dans l'éducation ou dans la formation professionnelle, dont la conception repose en grande partie sur le travail de pédagogues ? La création de tous ces projets industriels de Serious Games implique donc la participation d'experts du « domaine sérieux » qui est abordé à travers le jeu. Mais il ne faut pas non plus oublier les nombreux Serious Games de type « militant », créés par des amateurs désireux d'exprimer leur opinion sur un sujet donné. A mi-chemin entre amateurs et professionnels, des enseignants conçoivent parfois des jeux qu'ils utiliseront avec leurs élèves, quand ils ne proposent pas directement aux élèves de créer des jeux dans le cadre d'une activité pédagogique. Tous ces exemples illustrent une situation de plus en plus répandue dans le secteur du Serious Game : **de nombreuses personnes, qui ne possèdent aucune compétence particulière dans la création vidéoludique mais qui sont expertes d'un « sujet sérieux », manifestent l'envie de créer des Serious Games**. Nous qualifierons ces personnes de « novices ». Pour permettre à ces novices de réaliser des Serious Games, il convient de leur en faciliter la création. Cette thématique est actuellement au coeur de plusieurs projets de recherche *(p.39)*. Nos travaux de recherche s'inscrivent dans la continuité de ces projets, et s'attachent à réfléchir sur la problématique suivante :

Quelles sont les approches permettant de faciliter la création de Serious Games ?

Comme nous l'avons évoqué en début d'introduction, l'industrie du jeu vidéo de divertissement a déjà réfléchi à une question similaire : elle permet à certains de ses joueurs de créer des jeux vidéo sans posséder de compétences dans le domaine. **Peut-être qu'au sein des outils mettant la création vidéoludique à la portée d'amateurs se trouvent des approches qui seraient pertinentes pour le secteur du Serious Game ?** Mais nous pouvons également supposer que la création de Serious Games se distingue de celle de jeux vidéo de divertissement. Avant d'étudier le potentiel des outils *« Game Design »* accessibles aux amateurs pour la création de Serious Games, il convient donc d'étudier **comment la création d'un Serious Game se différencie de celle d'un jeu vidéo de divertissement ?** Notre problématique de recherche s'articule donc autour de deux questions complémentaires :

1) Quelles sont les spécificités de la création de Serious Games ?
2) Quelles sont les approches, issues du Game Design, permettant de la faciliter ?

3 CONTEXTE DE TRAVAIL

Afin de tenter d'apporter une réponse aux deux questions induites par notre problématique, plusieurs types de travaux sont nécessaires. D'une part, il semble important d'analyser les différentes approches mobilisées par les jeux vidéo de divertissement pour mettre leur création à la portée de novices, puis d'éprouver ces approches pour la création de Serious Games. Il semble également primordial d'étudier en détail la conception de Serious Games, par exemple dans un cadre professionnel, afin d'identifier les éventuelles spécificités de leur création par rapport au jeu vidéo de divertissement. Le travail de recherche présenté dans cette thèse s'inscrit donc dans le cadre d'une convention *CIFRE (Conventions Industrielles de Formation par la REcherche)* réunissant plusieurs partenaires :

- La ludothèque du centre culturel ***Odyssud***, rattachée à la mairie de ***Blagnac***. La ludothèque souhaitait réfléchir globalement à la place du jeu vidéo dans le domaine de la culture. Nous avons donc été amené à participer à de nombreuses animations culturelles relatives au jeu vidéo, que ce soit la présentation d'un Serious Game au grand public *(p.132)*, la mise en place d'un concours de création de jeu vidéo *(p.250)*, ou encore la réalisation d'un « jeu en réalité alternée », type de jeu novateur mélangeant plusieurs supports pour raconter une fiction *(p.291)*.

- La ***SEM Constellation***, aménageur de l'écoquartier *Andromède*, situé sur les communes de ***Blagnac*** et de ***Beauzelle***, dans la banlieue toulousaine. Ce partenaire souhaitait s'appuyer sur le Serious Game pour communiquer sur son projet urbain. Un des projets « industriels » de cette thèse fut donc la conception et la réalisation de Serious Games présentant de manière pédagogique l'écoquartier *Andromède* *(p.115)*.

- La société ***OKTAL***, spécialisée dans la réalisation de simulations destinées à la formation professionnelle. Cette entreprise souhaitait explorer l'apport potentiel des approches du *« Serious Game »* pour ses simulations. Avec ce partenaire, Nous avons travaillé sur la conception d'un Serious Game destiné à l'apprentissage de l'écoconduite : *geDriver* *(p.86)*. Nous avons également participé à une réflexion de fond sur les évolutions de l'outil technique interne utilisé pour réaliser ce jeu *(p.215)*.

Cette convention *CIFRE* nous a permis d'étudier de près le processus de création de Serious Games, tout en nous confrontant à des situations concrètes illustrant notre problématique. Ces retours d'expériences ont nourri notre travail de recherche, qui s'est appuyé sur l'analyse théorique des approches existantes. Au-delà de cette dimension « industrielle », nous avons également exploré un autre contexte d'application de notre problématique : la pédagogie. Nous nous appuyons pour cela sur des cours que nous dispensons auprès d'étudiants *(p.240)*.

4 ORGANISATION DE LA THESE

Afin d'essayer d'apporter une réponse à notre problématique, cette thèse consigne notre travail de recherche en quatre parties. La première d'entre elles *(p.17)* vise à définir les principaux domaines qui seront étudiés dans cette thèse : les *« Serious Games »* et le *« Game Design »*. Nous commencerons par présenter le « Serious Game » de manière générale *(p.18)*, avant d'aborder plus en détail les tendances actuelles du secteur en lien avec notre problématique *(p.39)*. Nous présenterons ensuite le domaine du « Game Design », autrement dit sur la conception de jeux *(p.42)*.

Suite à cette première partie définissant la manière dont nous aborderons ces deux domaines, nous les analyserons de manière plus poussée en essayant de les « croiser » en regard de notre problématique. Pour cela, nous étudierons l'aspect « théorique » dans la deuxième partie *(p.58)*, en analysant les outils théoriques de conception destinés au jeu vidéo de divertissement *(p.59)* et au Serious Game *(p.86)*, ainsi que les spécificités relatives à la conception de ces derniers *(p.106)*. La troisième partie sera consacrée à l'aspect technique *(p.155)*, à travers l'étude de la pertinence de plusieurs catégories de logiciels pour la création de Serious Games : les outils de « modding » *(p.163)*, les « usines à jeux » *(p.166)*, le « Jeu 2.0 » *(p.200)* et les outils explicitement destinés aux Serious Games *(p.214)*.

Enfin, dans la quatrième partie nous utiliserons ces résultats pour un exemple d'application concrète illustrant notre problématique : la création de Serious Games par des apprenants novices en la matière *(p.234)*. Afin d'utiliser le *« Serious Game Design »* comme méthode pédagogique, un enseignant doit mettre en place des approches permettant à ses étudiants de créer de tels jeux vidéo. Après avoir constaté, grâce à ce contexte, les limites des approches identifiées dans cette thèse *(p.248)*, nous réfléchirons aux solutions à développer en réponse à ces limites *(p.256)*. Cette réflexion nous amènera à travailler sur le développement d'un nouveau logiciel facilitant la création de Serious Games. *(p.296)*

[PARTIE I]
SERIOUS GAMES ET GAME DESIGN

Cette première partie de la thèse vise à préciser les deux domaines que nous étudierons lors de notre travail de recherche : le secteur du *« Serious Game »* et le champ du *« Game Design »*.

Nous présenterons tout d'abord notre contexte de recherche, le *« Serious Game »*, en cherchant notamment à le définir *(p.18)*, avant d'étudier d'autres projets de recherche qui s'inscrivent dans une problématique similaire à la nôtre *(p.39)*. Nous préciserons alors les définitions et concepts relatifs au champ du *« Game Design »* que nous mobiliserons durant nos travaux *(p.42)*. Nous passerons également en revue quelques approches de facilitation de la création vidéoludique spécifique à ce champ *(p.54)*.

La présentation croisée de ces deux domaines permettra de définir notre périmètre de recherche tout en précisant notre méthodologie de travail *(p.55)*.

INTRODUCTION AU SERIOUS GAME

Ce premier chapitre de la thèse vise à présenter notre contexte de recherche : le *« Serious Game »*. Nous commencerons par un retour sur la définition de ce terme et la classification des objets auxquels il renvoie. Nous présenterons ensuite une rapide « vue d'ensemble » du secteur à l'aide d'une analyse quantitative menée sur une large sélection de Serious Games. Nous aborderons enfin les tendances actuelles du secteur qui sont liées à notre problématique.

1 DEFINIR LE SERIOUS GAME

De prime abord, l'expression *« Serious Game »* apparaît comme un « oxymore »[1]. Mais les notions de « jeu » et de « sérieux » sont-elles finalement si opposées ?

1.1 ORIGINES DE L'OXYMORE « SERIOUS GAME »

Il semblerait que nous puissions faire remonter les origines de l'expression *« Serious Game »* à l'époque de la Renaissance *(XVe siècle)*. En Italie, les néoplatoniciens d'alors utilisaient l'expression *« serio ludere »* pour désigner le recours à l'humour dans la littérature afin de transmettre des notions sérieuses (Manning, 2004). Au-delà de ce « jeu linguistique », un autre exemple littéraire nous vient du roman *Den allvarsamma leken* de ***Hjalmar Södeberg***, dont le titre anglais est *The Serious Game* (Soderberg, 2001). Ecrit en 1912, ce classique de la littérature suédoise traite de l'adultère. Nous rencontrons ici l'idée de la mise en scène d'un jeu qui n'est pas « frivole » et possède des conséquences sur la vie réelle. Dans ce contexte, l'expression « Serious Game » s'inscrit donc dans une rupture avec la définition du « jeu » proposée par ***Huinzinga*** (1951), qui l'entend comme :

> *« [U]ne activité libre clairement séparée de la vie 'ordinaire' de par sa nature 'non sérieuse', mais qui en même temps absorbe le joueur de manière intense et totale »*[2]

Pourtant, il existe bien des jeux qui ne sont pas *« séparés de la vie ordinaire »*, à l'image de la pratique professionnelle d'un jeu ou d'un sport. L'expression « Serious Game » peut alors être utilisé pour qualifier une telle pratique professionnelle, comme le fait le joueur de cricket ***Mike Harfield*** (2008) dans son ouvrage *Not Dark Yet: A Very Funny Book About a Very Serious Game*. Nous pouvons alors nous référer à une célèbre citation de ***Freud*** (1985) :

> *« L'opposé du jeu n'est pas le sérieux, mais la réalité »*

Ainsi les notions de « jeu » et de « sérieux » ne seraient donc pas si contradictoires qu'on pourrait le penser au premier abord. Cela nous amène à la définition de la notion de *« Serious Game »* formalisée dans l'ouvrage *Serious Games* (Abt, 1970). Son auteur, ***Clark Abt***, est un chercheur américain (Abt Associates, 2005) ayant travaillé sur la conception de jeux tels que *T.E.M.P.E.R.* ***(Raytheon, 1961)***. Il s'agit d'un jeu de simulation sur ordinateur utilisé par des officiers de l'armée pour étudier le conflit de la « guerre froide » à l'échelle mondiale (Abt, Hodder, & O'Sullivan, 1965). Mais dans son ouvrage, ***Abt*** propose également de nombreux

[1] *« figure de style visant à rapprocher deux termes que leur sens devrait éloigner »*, d'après la définition relevée le 18-08-11 sur http://fr.wikipedia.org/wiki/Oxymore

[2] *"a free activity standing quite consciously outside 'ordinary' life as being 'not serious', but at the same time absorbing the player intensely and utterly"*

exemples de jeux sur des supports « non-informatique », tels que des jeux traitant des mathématiques ou des sciences humaines destinés à être utilisés dans un cadre scolaire. Son approche du « Serious Game » est particulièrement précise. Elle se démarque à ses yeux des pratiques « sérieuses » du jeu que nous venons d'évoquer :

> *« Les jeux peuvent être joués de manière sérieuse ou en dilettante. Nous considérons comme serious games les jeux explicitement et intentionnellement conçus à des fins éducatives, et non ceux principalement destinés au divertissement. Cela n'implique aucunement que les serious games ne soient pas, ou ne doivent pas, être amusants. »*[3]

Un autre exemple de jeu sur « support traditionnel » explicitement présenté comme un « Serious Game » se trouve dans l'ouvrage *The New Alexandria simulation: a serious game of state and local politics* (Jansiewicz, 1973). Cet ouvrage explique comment jouer à une simulation ludique des mécanismes de la politique intérieure américaine. Malgré son ancienneté, ce jeu est encore utilisé de nos jours pour enseigner, grâce à plusieurs rééditions depuis 2004. Notons au passage que l'enseignant à l'origine de ce jeu l'a volontairement conservé dans un format « non-informatique », car il pense que seules les interactions humaines sont à même de simuler la complexité de la politique (Jansiewicz, 2011). Et en effet, une étude (Kahn & Perez, 2009) sur l'utilisation de ce jeu en classe montre sa pertinence pour améliorer la compréhension du sujet par des étudiants assistant à un cours *« d'introduction au système politique américain »*.

L'expression « Serious Games » fut également utilisé dans le domaine de l'art et la culture. De 1996 à 1997, la britannique ***Barbican Art Gallery*** a accueilli une exposition intitulée *Serious Games: Art, Interaction, Technology* (B. Graham, 1996). L'ouvrage accompagnant cette exposition présente les travaux de huit artistes qui ont cherché à établir des connexions entre les jeux vidéo et l'art contemporain. L'œuvre d'une de ces artistes, ***Regina Corwell***, a été explicitement créée pour questionner le potentiel du jeu vidéo en tant que vecteur d'expression artistique :

> *« Si nous pouvons remplacer le côté « fun » des jeux, avec leurs messages explicites et implicites sur le pouvoir, la vitesse, la maîtrise et le contrôle, par des messages tels que ceux diffusés de manière opportuniste par les militaires ou par la culture du rendement associée au monde du travail et à la société de consommation, est-ce que l'art et la culture sont véritablement prêts à s'attaquer à une mosaïque aussi complexe ? »*[4]

Réalisant le fait que les jeux vidéo sont plus qu'un « simple divertissement » car ils diffusent parfois des messages liés aux domaines de la défense, de la formation professionnelle et de la publicité, cette artiste semble alors lancer un appel pour que le jeu vidéo soit associé à d'autres finalités, en particulier l'expression artistique.

Ce dernier exemple limite l'envergure des « Serious Games » aux seuls jeux vidéo, ce qui semble être le cas de la plupart des définitions actuelles (J. Alvarez, 2007; Michael & Sande Chen, 2005; Zyda, 2005). Toutes ces définitions ont visiblement été influencées par l'approche de ***Ben Sawyer*** et de son livre blanc intitulé *Serious Games: Improving Public*

[3] *"Games may be played seriously or casually. We are concerned with serious games in the sense that these games have an explicit and carefully thought-out educational purpose and are not intended to be played primarily for amusement. This does not mean that serious games are not, or should not be, entertaining"*

[4] *"If we shift from the fun of games with their overt or covert messages about power, speed, command and control to those same messages delivered for expediency and with urgency by the military and to the efficiency of the office workplace and the various heritage in consumer culture, are art and culture ready to squarely face this complex mosaic?"*

Policy through Game-based Learning and Simulation (Sawyer & Rejeski, 2002). Comme son titre le laisse supposer, ce livre blanc est un appel à l'utilisation du savoir et des technologies issues de l'industrie du jeu vidéo de divertissement pour améliorer les simulations et autres outils d'apprentissage utilisés dans les institutions publiques. Cependant, à l'exception de son titre, ce livre blanc ne fait jamais mention de l'oxymore « Serious Game ». A l'origine, ***Sawyer*** avait simplement intitulé son travail *Improving Public Policy through Game Based Learning and Simulations*. Son collègue ***David Rejeski*** trouvait cependant que ce titre n'était pas très engageant. Heureuse coïncidence, ce dernier venait de terminer la lecture d'un ouvrage intitulé *Serious Play* (Schrage, 1999), qui explique comment des compagnies privées s'appuient sur des simulations afin d'accompagner leur processus d'innovation. En référence à cet ouvrage, ***Rejeski*** modifia le titre du livre blanc rédigé par ***Sawyer*** pour y inclure l'oxymore « Serious Games ». Ce livre blanc fut rapidement suivi par la création de la *Serious Games Initiative*, une association fondée par les deux hommes afin de promouvoir l'usage du jeu vidéo à des fins sérieuses. Ainsi, l'oxymore « Serious Game » commençait à gagner en popularité auprès de nombreuses personnes (Sawyer, 2009). Autre coïncidence, quelques mois après la publication de ce livre blanc, l'armée américaine lança publiquement le jeu vidéo *America's Army* ***(U.S. Army, 2002)***. ***Sawyer*** déclarera alors (Gudmundsen, 2006) :

> *« [America's Army] est le premier Serious Game réussi et de qualité a avoir massivement attiré l'intérêt du grand public »*[5]

La combinaison du succès populaire d'*America's Army* avec les efforts de ***Sawyer*** et ***Rejeski*** nous invitent à définir l'année 2002 comme le point de départ de la « vague actuelle » des Serious Games. Ainsi, dans les années qui ont suivi, de nombreux jeux vidéo ont été réalisés pour servir des finalités sérieuses *(p.32)*. Ils ont été accompagnés par plusieurs tentatives de définition des « Serious Games ». La plus simple est sans aucun doute celle proposée par les concepteurs de jeux vidéo ***Chen & Michael*** (2005) :

> *« Tout jeu dont la finalité première est autre que le simple divertissement »*[6].

Dans le même temps, ***Zyda*** (2005), un chercheur américain ayant participé au développement d'*America's Army*, propose une définition plus spécifique :

> *« Un défi cérébral contre un ordinateur impliquant le respect de règles précises, et qui s'appuie sur le divertissement pour atteindre des objectifs liés à la formation institutionnelle ou professionnelle, l'éducation, la santé, la politique intérieure et la communication »*[7].

Ces tentatives ont également conduit ***Sawyer*** (2007) à affiner sa propre définition :

> *« Toute utilisation pertinente des technologies issues de l'industrie du jeu vidéo à des fins autres que le divertissement »*[8]

Enfin, de son côté le chercheur français ***Alvarez*** (2007) en propose une plus complète :

> *« Application informatique, dont l'intention initiale est de combiner, avec cohérence, à la fois des aspects sérieux (Serious) tels, de manière non exhaustive et non exclusive, l'enseignement, l'apprentissage, la communication, ou encore l'information, avec*

[5] *"[America's Army] was the first successful and well-executed serious game that gained total public awareness"*
[6] *"Games that do not have entertainment, enjoyment or fun as their primary purpose"*
[7] *"A mental contest, played with a computer in accordance with specific rules, that uses entertainment, to further government or corporate training, education, health, public policy, and strategic communication objectives"*
[8] *"Any meaningful use of computerized game/game industry resources whose chief mission is not entertainment"*

> *des ressorts ludiques issus du jeu vidéo (Game). Une telle association, qui s'opère par l'implémentation d'un "scénario pédagogique", qui sur le plan informatique correspondrait à implémenter un habillage (sonore et graphique), une histoire et des règles idoines, a donc pour but de s'écarter du simple divertissement. Cet écart semble indexé sur la prégnance du "scénario pédagogique". » [p.51]*

Ces diverses définitions semblent toutes être porteuses de la même volonté : **mettre en relation une finalité sérieuse avec des savoirs et des technologies issues de l'industrie vidéoludique**. Cependant, certains acteurs ne procèdent pas ainsi. Par exemple, dans le secteur de la formation professionnelle, des formateurs s'appuient sur des jeux de rôle ou de plateau plutôt que sur du jeu vidéo. Le *Centre International de la Pédagogie d'Entreprise (CIPE)*[9] est un exemple de ces acteurs. Bien que dirigeant un studio de création de Serious Games basés sur le jeu vidéo, ***Kevin Corti*** (2007) a publié un article très critique sur ce sujet. Il appelle à l'élargissement des définitions usuelles du Serious Game en rappelant que certains acteurs de cette industrie ne se reconnaissent pas dans cette expression, et lui préfèrent d'autres appellations telles que *Game-Based Learning* ou *Simulation*. Ainsi, si aujourd'hui le lien avec les jeux vidéo semble être majoritaire dans l'industrie du Serious Game, il n'est pas la seule voie possible. Le grand nombre de termes utilisés pour désigner un *« jeu dont la finalité première est autre que le simple divertissement »* en témoigne (Sawyer & Peter Smith, 2008) :

> *Educational Games, Simulation, Virtual Reality, Alternative Purpose Games, Edutainment, Digital Game-Based Learning, Immersive Learning Simulations, Social Impact Games, Persuasive Games, Games for Change, Games for Good, Synthetic Learning Environments, Games with an Agenda…*

Cette multitude d'appellations reflète autant le nombre conséquent d'acteurs qui s'intéressent au Serious Game que la diversité de leurs approches. En effet, sans pour autant proposer une analyse et un point de vue véritablement formalisés, chaque acteur possède une vision propre du « Serious Game » selon son secteur d'origine : *enseignant, concepteur de simulation professionnelle, éditeur de jeu vidéo, groupe médiatique…* Cette diversité des acteurs nous conduit à aborder une question fondamentale : quelles sont les réalités recouvertes par la notion de « sérieux » dans l'expression « Serious Game » ? Dans les définitions que nous venons d'étudier, la notion de « sérieux » semble renvoyer aussi bien à des finalités publicitaire, pédagogique, médicale, sociale voire même artistique. Il semble pourtant qu'il s'agisse là de finalités très différentes. Comment rapprocher la diffusion de messages marketing avec la transmission de savoirs permettant de développer l'esprit critique ? Comment associer l'entraînement à la pratique de gestes chirurgicaux avec un commentaire sur l'actualité internationale ? C'est pourtant ce que fait la notion de « sérieux » dans l'expression « Serious Game ». Cette tendance à considérer comme « sérieux » toute finalité qui est autre que le fait de procurer un amusement s'observe aussi bien dans les définitions que nous venons d'étudier que lors de l'exploration des jeux explicitement présentés comme « Serious Games » *(p.32)*.

En tant que chercheur, nous pouvons être réservés quant à la pertinence d'un champ d'application aussi large de l'expression « Serious Game ». Nous pourrions par exemple tenter de l'affiner en remplaçant la notion de « sérieux » par la notion « d'utilité », **proposant ainsi de traduire l'expression « Serious Game » non pas comme « jeu sérieux » mais par l'expression « jeu à vocation utilitaire »**. Si cette expression apparaît comme plus précise, elle n'est pourtant pas employée au sein de notre champ d'étude. Dans cette thèse, nous continuerons donc à employer l'expression « Serious Game », associée à la notion de

[9] http://www.cipe.fr/

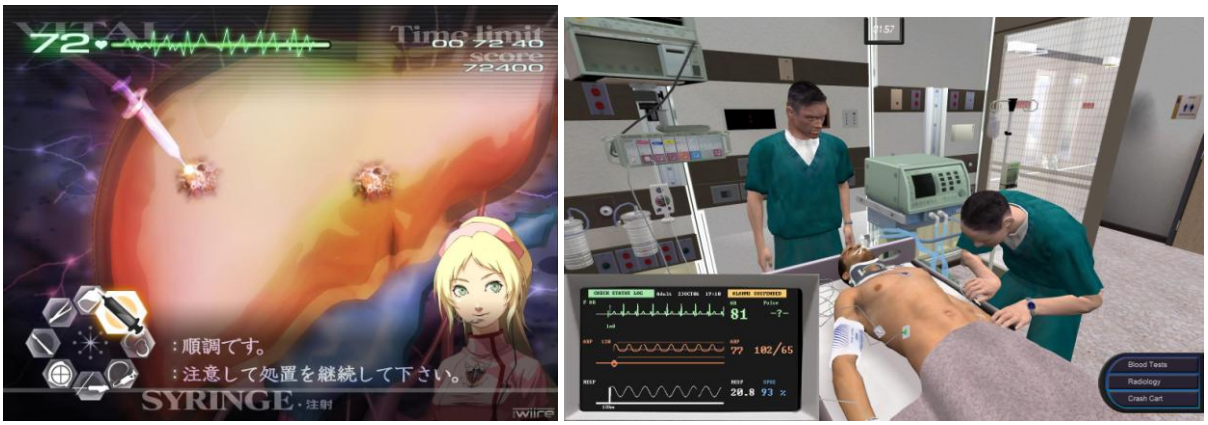
« sérieux », tout en précisant ici que cette dernière renvoie à une collection de finalités utilitaires disparates. Pour plus de simplicité, nous désignerons cet ensemble de finalités utilitaires disparates par l'expression **« dimension sérieuse »**. Au-delà de la question du périmètre de la notion de « sérieux », nous avons également constaté que les différents acteurs du champ des « Serious Games » ont parfois des divergences sur la notion de « jeu ». Moins profonde, ces dernières semblent s'en tenir au support de jeu utilisé. Nous proposons alors d'utiliser l'expression **« dimension ludique »** pour renvoyer aux différentes formes de jeux recouvertes par l'expression « Serious Game ». Cela nous permet de synthétiser d'une manière simple sa définition :

Nous pouvons considérer que **le « Serious Game » est un objet mélangeant deux dimensions : une « dimension sérieuse », renvoyant à tout type de finalité utilitaire, et une « dimension ludique », correspondant à un jeu matérialisé sur tout type de support.**

1.2 DIFFERENCE ENTRE SERIOUS GAME ET JEU VIDEO : NOTION DE SERIOUS GAMING

Si, par définition, un « Serious Game » peut être lié à tout type de support, dans le cadre de cette thèse nous nous consacrerons uniquement à ceux sur support informatique. Comme nous venons de le voir, ces derniers caractérisent la « vague actuelle » des Serious Games. Cela nous amène alors à nous interroger : qu'est-ce qui différencie fondamentalement un Serious Game d'un jeu vidéo ?

De prime abord, *Trauma Center : Second Opinion* ***(Atlus, 2006)*** pourrait être considéré comme un Serious Game lié à la santé, étant donné que ce jeu propose d'incarner un chirurgien pratiquant de nombreuses opérations. Pourtant, au-delà du contexte hospitalier et de quelques références aux pratiques médicales réelles, ce titre ne propose aucune dimension « sérieuse ». Ici, le logiciel a clairement été conçu pour divertir l'utilisateur. Le domaine médical ne sert que de contexte au scénario « ludique ». A l'inverse, *Pulse!!* ***(Breakaway, 2007)***, permet de dispenser un entraînement autour des compétences cliniques requises pour faire face à plusieurs situations d'urgence, telles que des accidents de transport ou une attaque bioterroriste. Là où *Trauma Center* demande au joueur de détruire des virus à l'apparence de dragons en utilisant un laser sur le cœur des patients, *Pulse!!* le met face à des cas médicaux réels à élucider en utilisant des techniques médicales actuelles.



1. Le jeu vidéo *Trauma Center : Second Opinion* et le Serious Game *Pulse!!*

Cependant, rien n'empêche de jouer à *Trauma Center* en adoptant une posture « sérieuse ». Cela est également vrai pour tout jeu vidéo provenant de l'industrie du divertissement qui peut potentiellement servir à des fins tout à fait sérieuses a posteriori. De nombreux exemples sont à recenser dans le secteur de l'éducation comme nous le présentent, entre autres, les ouvrages de ***Prensky*** (2007), ***Gee*** (2007) ou encore ***Shaffer*** (2007). En France, le collectif ***Pedagame*** effectue des expériences de terrain sur l'utilisation de jeux vidéo de divertissement

à des fins pédagogiques. Par exemple, le jeu de karaoké *Singstar PS3* ***(SCE London Studio, 2008)*** est utilisé comme support de cours afin de travailler la prononciation de l'anglais avec des collégiens. Dans un autre registre, le jeu de « question-réponse » *Buzz! Quiz TV* ***(Relentless Software, 2008)*** est détourné par les enseignants en histoire et géographie afin de revenir sur des notions abordées en cours. A cette fin, ils s'appuient sur la possibilité de créer des questions personnalisées offerte par ce titre (Llanas, 2009). De son côté, le ***Réseau Ludus***, qui regroupe des enseignants utilisant tout type de jeux à des fins pédagogiques, rapporte également de nombreuses expériences d'utilisation de jeux vidéo de divertissement en classe. Un des enseignants fondateur de ce réseau, ***Yvan Hochet*** (2011), précise que l'enseignant à un rôle primordial pour ce genre de pratique :

> *« La simple mise à disposition d'un jeu ne semble pas changer grand-chose à ce que les élèves apprennent. […] La médiation de l'enseignant, qui pense la place du jeu dans une démarche d'apprentissage, reste donc indispensable. » [p.107]*

D'après cet enseignant, le jeu vidéo de divertissement peut s'avérer un formidable matériel pédagogique dans plusieurs situations. Il prend pour exemple ses propres expérimentations en la matière. Il utilise notamment *Sim City* ***(Maxis, 1989)*** pour enseigner la géographie, en montrant à ses élèves que les liens entre les différents facteurs de simulation d'une ville dans ce jeu sont loin de refléter la réalité. Du côté de l'histoire, il propose à ses élèves de gérer un château dans *Lords of the Realms II* ***(Impressions Games, 1996)***, avant de les inviter à rédiger un texte mettant en relation leur actions dans le jeu avec le contexte historique de l'époque.

2. Les jeux vidéo *Buzz! Quiz TV* et *Sim City*

Dans une logique similaire mais appliquée au domaine de la santé, l'exemple de ***Michael Stora*** (2005) est on ne peut plus significatif. A travers son ouvrage *Guérir par le virtuel*, ce psychologue raconte comment il utilise le jeu *ICO* ***(Sony CE Japon, 2001)*** en le détournant de sa finalité première de « simple divertissement ». Lors de séances thérapeutiques avec des enfants, il leur fait emprunter un passage précis de ce jeu. Ce passage demande au joueur de maintenir la main d'une princesse, en gardant un bouton enfoncé, afin de la guider vers la sortie. Cependant, une fois rendu à destination, le joueur doit lâcher ledit bouton et laisser la princesse s'en aller. Certains enfants refusent cette situation et sont désorientés. Le thérapeute tente alors d'établir le dialogue en transposant le vécu familial de l'enfant au travers de la situation présentée par le jeu.

3. Le jeu vidéo *ICO* et le Serious Game *EarthQuake in Zipland*

Néanmoins, une différence fondamentale persiste entre ce type d'approche visant à détourner des jeux vidéo de divertissement et les Serious Games tels que définis précédemment. Si le résultat apparaît similaire (*un jeu utilisé à des fins sérieuses)*, seul le Serious Game a été explicitement conçu pour cet usage. Par exemple, *Earthquake in Zipland* ***(Zipland Interactive, 2006)*** se présente sous la forme d'un jeu d'aventure qui met en scène une île frappée par un tremblement de terre. A la suite de cet événement, l'île est coupée en deux. Le père et la mère du jeune héros se retrouvent isolés sur ces deux nouveaux territoires. Ils ne pourront plus jamais se retrouver ensemble. Destiné aux enfants, ce titre a pour vocation d'aborder le thème du divorce et de les inviter à s'exprimer sur le sujet. Les concepteurs de ce Serious Game ont donc prévu et anticipé un tel usage. Cette démarche se distingue de l'idée de prendre un jeu vidéo commercial pour lui assigner une nouvelle vocation a posteriori. Cet argument est logiquement mis en avant par les industriels du Serious Game pour valoriser leur savoir-faire, ce qui tend à exclure les approches de détournement du champ des « Serious Games ». Sans forcément répondre à cette délicate question, ***Jenkins & al.*** (2009) ont évoqué le fait qu'il serait plus pertinent de parler de *« Serious Gaming »* que de *« Serious Game »*, car un jeu ne peut avoir des conséquences « utilitaires » que lorsqu'un joueur l'utilise. Ainsi, en considérant la différence de conception entre les titres « détournés » et les autres, nous proposons de réserver l'expression *« Serious Game »* à des jeux qui ont été explicitement destinés à des *« finalités autres que le simple divertissement »* par leurs concepteurs. Nous proposons ensuite que l'expression *« Serious Gaming »* renvoie à toute utilisation d'un jeu à des fins autres que le simple divertissement, quelle que soit l'intention originelle de son concepteur. Le *« Serious Gaming »* englobe donc l'utilisation des *« Serious Games »* ainsi que les approches de « détournement vidéoludique », qui permettent à un jeu donné de servir des finalités sérieuses non anticipées par ses concepteurs.

Une dernière approche mérite enfin d'être mentionnée ici, car elle se trouve à mi-chemin entre la conception de Serious Game et le détournement de jeux destinés au divertissement. Il s'agit de la modification logicielle. Connue sous le nom de *« modding »*, le fait de modifier un jeu existant pour en diffuser une version différente est une pratique très répandue dans la culture vidéoludique *(p.51)*. La particularité d'un *« mod »* tient au fait qu'il n'est pas autonome, et nécessite de posséder le jeu de base pour fonctionner. Notons également que les concepteurs d'un *« mod »* n'ont généralement aucun lien direct avec ceux qui ont réalisé le jeu d'origine. Dans la plupart des cas, le *« modding »* se cantonne à la modification de jeux à des fins de divertissement. Mais il en existe certains qui transforment un jeu vidéo de divertissement en Serious Game. Par exemple, *Escape from Woomera* ***(Kate Wild et al., 2003)*** modifie le jeu *Half-Life* ***(Valve Software, 1998)***. Il s'appuie sur la structure ludique de ce dernier pour alerter l'opinion publique sur les conditions de vie dans un camp de réfugiés situé en Australie. Nous observons dans cet exemple la présence des deux dimensions « ludique » et « sérieuse » de manière explicite. Par rapport à la typologie évoquée précédemment, nous proposons donc

d'inclure cette approche dans le champ du *« Serious Game »*, car la présence des deux dimensions ne repose pas sur un détournement d'usage mais sur un choix de conception.

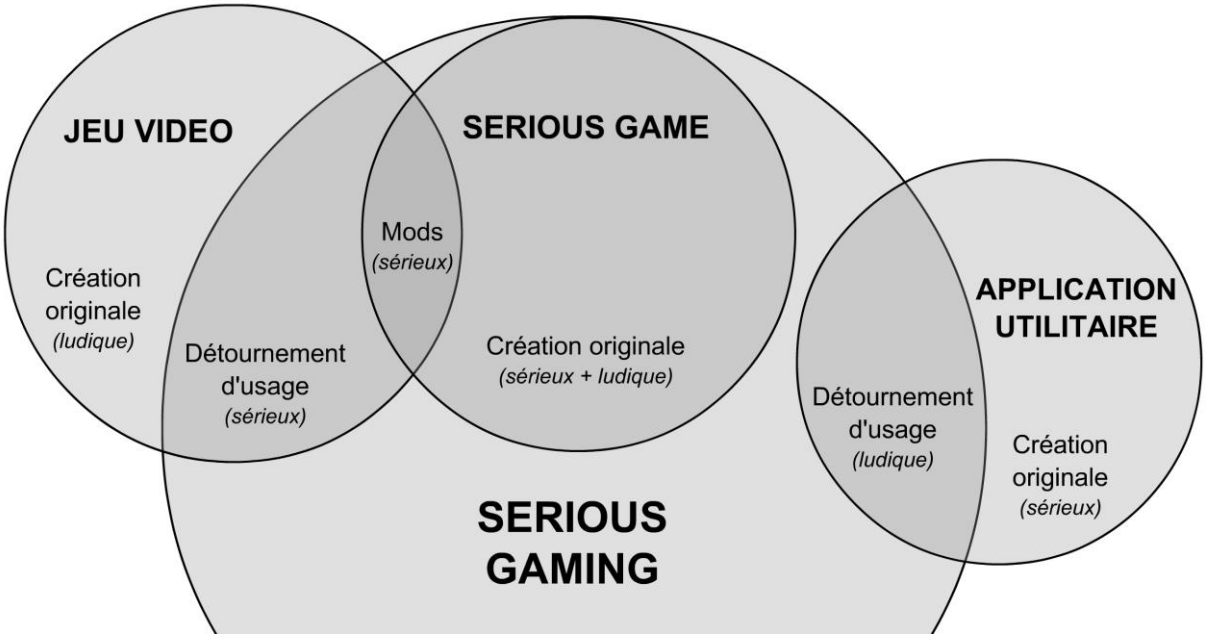
4. Le jeu vidéo *Half-life* et le mod *Escape from Woomera* qui le transforme en Serious Game

1.3 SYNTHESE : POSITIONNER LE SERIOUS GAME

Si nous distinguons les Serious Games des jeux vidéo dédiés au seul divertissement, nous devons également tenir compte de tous les logiciels qui ne présentent aucune dimension ludique : les applications utilitaires. Par exemple, *Medical Clinical Simulator* ***(Alfa Multimédia, 2008)*** permet d'étudier différents cas cliniques, basés sur de vrais dossiers. Mais il est également proposé à l'utilisateur d'établir son propre diagnostic. Cette partie de l'application peut être vue comme purement éducative, donc « sérieuse ». Cependant, d'autres personnes pourraient tout à fait y voir une approche ludique[10], rangeant alors *Medical Clinical Simulator* dans le champ du « Serious Game ». Au final, les frontières entre « Serious Game », « jeu vidéo » et « application utilitaire » sont poreuses car l'appréciation de la nature « ludique » ou « sérieuse » d'une application reste subjective. En cas de doute, nous proposons de se référer à l'approche du concepteur de l'application, qui, pour réaliser son logiciel, s'inscrit a priori dans une seule de ces catégories.

En conclusion nous pouvons proposer les définitions suivantes. Un *« Serious Game »* se caractérise par la présence d'une dimension « ludique » et d'une dimension « sérieuse » explicitement souhaitées par son concepteur. Cependant, les utilisateurs peuvent utiliser un jeu vidéo ou une application utilitaire d'une façon qui n'a pas été forcément prévue par son concepteur. Il s'agit alors d'un « détournement d'usage », qui permet par exemple d'utiliser à des fins sérieuses un jeu vidéo à la base conçu pour le divertissement. Ces deux approches, conception originale et détournement d'usage, définissent le *« Serious Gaming »*. Le schéma ci-dessous illustre les relations entre ces différentes notions :

[10] Voir par exemple : http://www.serious-game.fr/wordpress/index.php/269/quoi-de-neuf-docteur/ (le 11-07-11)



5. Relations entre les notions de Serious Game, Serious Gaming, jeu vidéo et application utilitaire

2 CLASSIFIER LE SERIOUS GAME

Une des meilleures manières d'appréhender un nouvel objet est de tenter de classifier ses différentes formes. En effet, l'élaboration d'un système de classification implique une réflexion particulièrement poussée sur la définition d'un objet, de manière à pouvoir mettre en évidence des caractéristiques qui permettent de le classifier. Nous proposons donc de passer en revue plusieurs systèmes de classifications applicables au « Serious Game », avant de présenter nos propres travaux en la matière.

2.1 UNE PLURALITE D'APPROCHES CLASSIFICATOIRES

En écho à la diversité des définitions existantes du Serious Game, plusieurs méthodes existent pour tenter de le classifier. A ce jour, aucun consensus ne semble s'être établi autour d'un système particulier, et la croissance constante de ce secteur laisse à penser que l'évolution des classifications n'est pas non plus prête de s'arrêter. Malgré la relative jeunesse de cette thématique de recherche, nous recensons déjà plusieurs approches bien distinctes.

2.1.1 LES CLASSIFICATIONS FONDEES SUR LES « MARCHES »

Historiquement, il semble s'agir du premier type de classification mobilisé dans le champ du « Serious Game ». Utilisant un seul critère, cette approche organise les Serious Games selon leurs secteurs d'applications, également dénommés « marchés » :

- ***Zyda*** (2005) définit explicitement les Serious Games pour cinq domaines d'application : *Healthcare, Public policy, Strategic Communication, Defense, Training & Education.*
- ***Chen & Michael*** (2005) classent les Serious Games selon huit marchés différents: *Military Games, Government Games, Educational Games, Corporate Games, Healthcare Games, Political Games, Religious Games, Art Games.*
- ***Alvarez & Michaud*** (2008) mettent en évidence sept marchés principaux pour les Serious Games: *Defense, Training & Education Games, Advertising, Information & Communication, Health, Culture, Activism.*

Bien que simple à utiliser, ces systèmes souffrent de deux limites majeures. Tout d'abord, la croissance du secteur du Serious Game implique l'apparition régulière de nouveaux « marchés », rendant leur liste de catégories rapidement obsolète. Ensuite, ces systèmes étant focalisés sur les secteurs d'applications, ils ne renseignent aucunement sur le type de finalité visée par chaque jeu.

2.1.2 LES CLASSIFICATIONS ANALYSANT LES « FINALITES »

En complément des classifications focalisées sur les « marchés », un autre type de classification à un seul critère permet de classifier les Serious Games selon leur « finalité » :

- ***Bergeron*** (2006) présente sept types de finalités dans son ouvrage: *Activism games, Advergames, Business Games, Exergaming, Health and Medicine Games, News Games, Political Games.*
- ***Despont*** (2008) propose une typologie regroupant quatre finalités pour les Serious Games: *Advert Games, Institutional Serious Games, Business Games, Learning Games.* Cette typologie est dérivée d'un autre système du même auteur qui recense six « intentions sérieuses » : *sensibiliser, simuler, former, informer, éduquer, influencer.*
- ***Alvarez & Rampnoux & al.*** (2007) ont également proposé six types de finalités pour les Serious Games: *Edugames, Advergames, Newsgames, Activism games, Edumarket games, Training & Simulation games.*

Bien qu'étant toujours fondées sur un critère unique, ces classifications se révèlent plus complexes à utiliser. Elles présentent des catégories hétérogènes qui semblent parfois liées au marché plutôt qu'à leur finalité, ce qui rend ambiguë la classification de certains titres. Mais surtout, elles ne renseignent aucunement sur les secteurs d'applications.

2.1.3 UNE TAXINOMIE A DOUBLE CRITERE

Face à ces deux types de classification monocritère, ***Sawyer & Smith*** (2008) proposent un système combinant les deux informations au sein d'une taxinomie à deux entrées :

- **Marché (Market):** *Government & NGO, Defense, Healthcare, Marketing & Communication, Education, Corporate, Industry.*
- **Finalité (Purpose):** *Games for Health, Advergames, Games for Training, Games for Education, Games for Science and Research, Production, Games as Work.*

Afin de permettre une analyse encore plus détaillée, chaque catégorie du critère *« Finalité »* possède également une « sous-taxinomie » propre, dont la complexité est très variable. La grande force de cette taxinomie est d'avoir su synthétiser les précédentes classifications en un seul système, certes plus complexe, mais qui permet enfin d'avoir une vision globale du champ des Serious Games. Cependant, elle souffre également de deux limites importantes. La première limite concerne ses différentes catégories du critère *« Finalité »*, dont les définitions pourraient être plus précises, comme nous le verrons ultérieurement *(p.30)*. La seconde limite est qu'il manque encore une information essentielle à cette classification. Nous avons vu lors de notre étude des définitions qu'un Serious Game repose sur l'association d'une dimension « sérieuse » et d'une dimension « ludique » *(p.18)*. Or ce système, comme toutes les classifications dédiées au Serious Game qui l'on précédées, ne prend en compte que la dimension « sérieuse », occultant totalement la dimension « ludique ». Pour tenter de compléter cette dernière, il convient donc à présent d'explorer les approches classificatoires des jeux vidéo de divertissement. Par cette démarche nous visons le recensement de paradigmes qui pourraient être importées dans le champ des Serious Games.

2.1.4 LES CLASSIFICATIONS SPECIFIQUES AU JEU VIDEO DE DIVERTISSEMENT

Si aucun système dédié aux Serious Games ne semble encore prendre en considération l'aspect ludique, le champ des jeux de divertissement est par contre riche en travaux dédiés à sa classification. Ces systèmes proviennent de sources aussi variées que les universitaires, les concepteurs, la presse spécialisée ou les joueurs eux-mêmes. Cependant, aucune classification de l'aspect ludique des jeux vidéo de divertissement ne semble encore faire consensus.

L'approche la plus courante pour classifier les jeux vidéo consiste à les organiser en « genres ». Plusieurs formes de classification par « genre » existent, la plus répandue étant celle dite « libre ». Comme ce nom laisse supposer, ces classifications naissent d'une réflexion subjective et empirique. En conséquence, elles sont incroyablement nombreuses, et bien qu'elles partagent une structure similaire, leurs définitions et choix de genres varient considérablement. Si les classifications « libres » représentent une part importante de la culture vidéoludique, elles ne sont malheureusement pas un système de classification fiable dans le temps (Letourneux, 2006). Face à cette masse de classifications « libres », plusieurs concepteurs ou universitaires ont essayé de proposer une classification par genre « stable », à travers une analyse critique de celles dites « libres ». Parmi ces classifications visant à améliorer celles émanant de sources empiriques, nous recensons entre autres les travaux de ***Crawford*** (1982), ***Myers*** (1990), ***Le Diberder*** (1993), ***Wolf*** (2002), ***Rollings & Adams*** (2003) et ***Natkin*** (2006). Ces systèmes de classification n'ont malheureusement amené aucun consensus, sûrement à cause de défauts latents mis en évidence par ***Letourneux*** (2006) et ***Apperley*** (2006). A travers leurs critiques des systèmes existants, ces deux auteurs appellent individuellement à explorer d'autres voies que les genres pour classifier les jeux vidéo. Il existe donc quelques approches différentes pour la classification de la dimension ludique, à l'image des travaux de ***Aarseth & al.*** (2003), ***Strange Agency*** (2006), ***Elverdam & Aarseth*** (2007), ou encore notre propre système (Djaouti, J. Alvarez, Jessel, Methel, & Molinier, 2008). Ces derniers exemples se focalisent tous sur la structure interne des jeux vidéo, avec un niveau de détail plus ou moins élevé, afin de les classifier.

En résumé, nous avons ici un certain nombre de systèmes classificatoires permettant de rendre compte de la dimension « ludique » d'un jeu vidéo. Par ailleurs, il existe un certain nombre de systèmes permettant de classifier les Serious Games selon leur dimension « sérieuse ». A partir de l'étude de ces sources, ne pourrions-nous pas imaginer un système permettant de classifier les Serious Games à la fois par leurs dimensions « ludique » et « sérieuse » ?

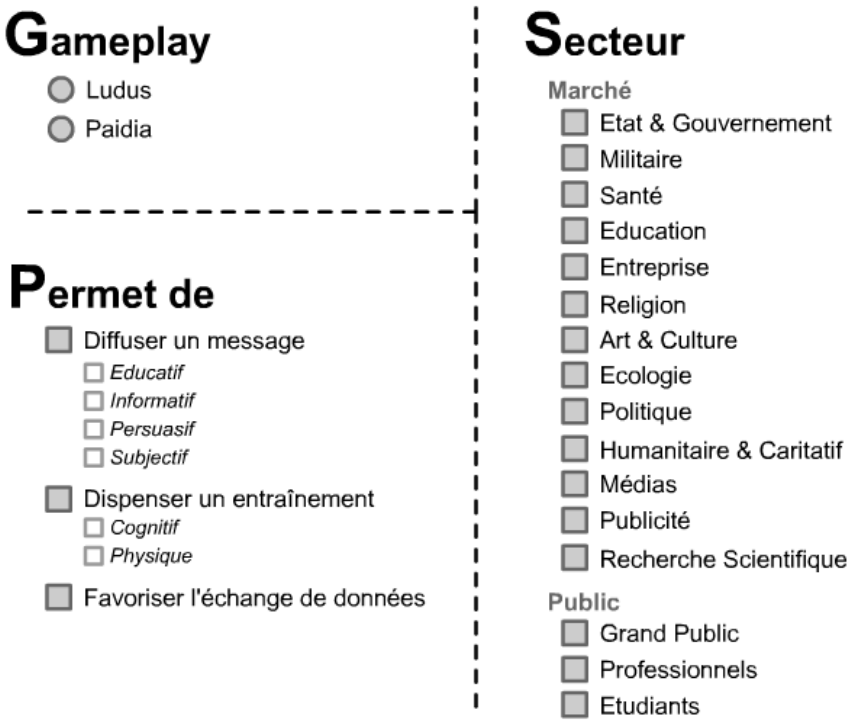
2.2 LE MODELE G/P/S DE CLASSIFICATION DES SERIOUS GAMES

Afin d'élaborer un système visant à classifier les Serious Games selon leurs dimensions « ludique » et « sérieuse » de manière simultanée, nous proposons d'utiliser trois types de critères :

- **Gameplay,** basé sur le *« gameplay »*[11] du Serious Game. Ce critère renseigne sur la dimension « ludique » en donnant des informations sur le type de structure utilisée pour créer le jeu.
- **Permet de *(Purpose),*** basé sur la finalité du Serious Game. Ce critère renseigne sur la ou les *« vocations dépassant le simple divertissement »* souhaitées par le concepteur.
- **Secteur *(Scope),*** basé sur les domaines d'applications visés par le « Serious Game ». Ce critère informe sur le type de public *(marché, âge…)* que le concepteur cherche à atteindre.

[11] Terme anglophone ne possédant pas de traduction française directe. Historiquement, ce mot est dérivé de l'expression *« How the game plays ? »*, qui était le titre des fiches d'instructions se trouvant sur les premières bornes d'arcade. Aujourd'hui, ce terme renvoie généralement au principe de jeu ou à des notions connexes.

Ce *modèle G/P/S* est un guide qui permet de classifier les Serious Games à la fois par leur dimension « ludique » *(Gameplay)*, et leur dimension « sérieuse » *(Permet de & Secteur)*. Afin d'appliquer ce modèle classificatoire, une spécification précise de critères a été posée pour chacun des trois aspects. Ces critères sont synthétisés par le schéma ci-dessous :



6. Les différents critères du *modèle G/P/S* de classification des Serious Games

2.2.1 CRITERES DE CLASSIFICATION DE L'ASPECT « GAMEPLAY »

Généralement, les jeux vidéo sont classifiés en regard de leur principe ludique par des « genres » vidéoludiques. Si cette notion de genre est une part importante de la culture du jeu vidéo, des travaux ont pointé ses limites en tant que système de classification fiable dans le temps *(p.28)*. En considérant ces limites, nous avons choisi de classifier le gameplay par une approche simple, destinée à rendre compte de la nature générale de la structure ludique.

Introduites par ***Caillois*** (1967), puis actualisées par ***Frasca*** (2003), les notions de *« paidia »* et de *« ludus »* font état de deux formes ludiques distinctes. Leur différence se situe sur la construction de la structure ludique. Par exemple, *Sim City* ***(Maxis, 1989)*** semble tenir de la *« paidia »*, car il ne propose pas d'objectifs explicites à atteindre permettant au joueur de « gagner ». Selon les définitions proposées par ***Salen & Zimmerman*** (2003), *Sim City* est en effet un jeu dépourvu de *« quantifiable outcome »*, un état final mettant fin à la partie tout en proposant une évaluation de la performance du joueur[12]. A l'inverse, un jeu *« ludus »* comme *Pac-man* ***(Namco, 1980)*** définit des objectifs explicites *(manger toutes les pastilles en évitant les fantômes)* qui sont utilisés pour évaluer la performance du joueur, à travers un retour positif *(gain de points de score)* ou négatif *(perte d'une vie)*. Cette distinction entre *« ludus »* et *« paidia »* s'applique également aux Serious Games. Par exemple, *Stop Disasters!* ***(Playerthree, 2007)*** propose une structure de type *« ludus »*, alors que *September 12th* ***(Gonzalo Frasca, 2003)*** s'appuie sur le modèle *« paidia »*. Notons que la différence entre *« paidia »* et *« ludus »* équivaut à celle que l'on retrouve entre *« play »* et *« game »* dans la

[12] Pour *Sim City*, il est certes possible d'imaginer une forme d'évaluation de la performance du joueur à partir de facteurs tels que le nombre d'habitants ou le budget restant. Cependant, ce sera bien au joueur de se fixer des critères d'évaluation, contrairement à un jeu *« ludus »* où ces derniers sont définis par le concepteur du jeu.

langue anglaise, tel qu'expliqué par ***Brougère*** (2005). *« Play »* se rapproche de l'idée d'amusement *(« paidia »)* alors que *« game »* sous-tend la notion de règles de jeu *(« ludus »)*.

En plus de différencier les Serious Games selon qu'ils soient basés sur une structure *« paidia »* ou *« ludus »*, nous essayons de rendre compte des principales règles de jeu employées par le concepteur. Pour cela, nous nous appuyons sur notre système de *« briques de GamePlay »*, qui est détaillé dans la dernière partie de cette thèse *(p.267)*. Pour faire simple, ce système propose dix « règles de base » que l'on retrouve couramment dans les jeux vidéo. Ces « règles de bases » sont ensuite associées à un « verbe », formant ce que nous appelons une « brique ». Nous différencions les « briques » liées aux objectifs proposés au joueur de celles qui sont liées aux divers moyens et contraintes mis à sa disposition pour essayer de les atteindre :

- **Objectifs à atteindre:** *Eviter, Atteindre, Détruire.*
- **Moyens & Contraintes:** *Créer, Gérer, Déplacer, Choisir, Tirer, Ecrire, Aléatoire.*

Le gameplay de chaque Serious Game est alors retranscrit par une combinaison comprenant une à dix de ces « briques », donnant ainsi une idée générale de sa structure ludique de base.

2.2.2 CRITERES DE CLASSIFICATION DE L'ASPECT « PERMET DE »

L'appréciation des finalités qu'un concepteur souhaite viser à travers la réalisation d'un Serious Game est loin d'être une tâche simple. Usuellement, différentes catégories telles que *Advergames, Edugames, Exergames, Datagames, News games, Edumarket games, Health games, Military games…* sont employées pour distinguer les différents « types de finalité » d'un Serious Game *(p.27)*. À nos yeux, l'emploi de ces catégories n'est pas forcément des plus pertinent, car elles souffrent des mêmes limites que celles des « genres vidéoludiques » pour la classification du gameplay *(p.28)*. Nous avons donc essayé d'établir une liste de catégories synthétique dans l'espoir qu'elle soit plus fiable dans le temps.

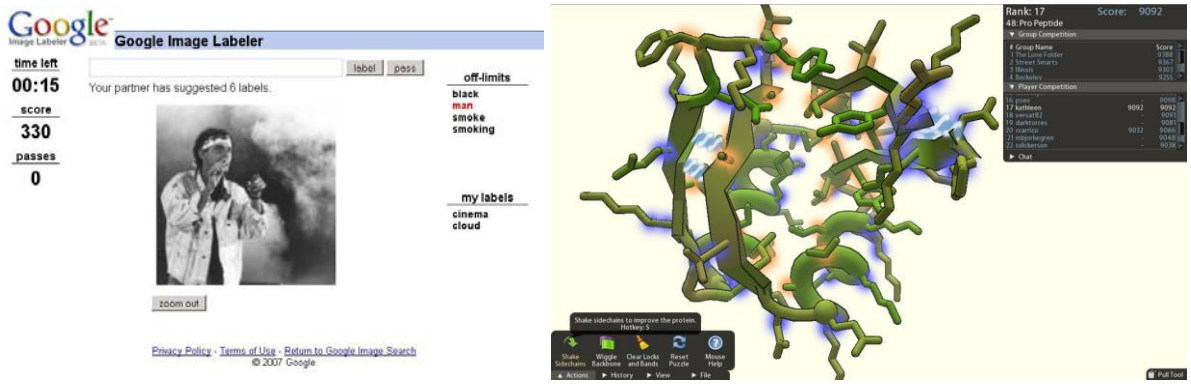
Parmi les catégories utilisées dans les classification précédentes pour décrire la finalité d'un Serious Game *(p.27)*, nous trouvons *« Edugames » (et ses équivalents « Games for Education » et « Learning Games »)* et *« Advergames » (et son équivalent « Advert Games »)*. D'une manière simple, un *« Edugame »* permet de transmettre un message éducatif. Un *« Advergame »* permet de promouvoir un produit ou service, que l'on peut interpréter comme la transmission d'un message volontairement positif à propos dudit produit ou service. En quelque sorte, bien que leur intention soit différente *(commerciale ou pédagogique)*, ces deux catégories de Serious Games semblent avoir pour finalité de **diffuser un « message »**. Une observation similaire peut être menée sur les autres catégories usuelles : les *« Newsgames »* diffusent un message informatif, les *« Political Games »* un message politique… Au final, les catégories de « finalité » généralement utilisées servent apparemment à différencier la nature du message diffusé par un Serious Game. Cette caractérisation de la « nature » des messages est ici liée au secteur d'application. Or, le *modèle G/P/S* possède déjà un critère *« Secteur »* qui permet de les référencer *(p.32)*. Il semble donc plus pertinent de classifier les messages par leur nature. Nous proposons de les recenser comme suit :

- Le message **informatif**, visant à diffuser un point de vue neutre.
- Le message **éducatif**, visant à transmettre un savoir ou un enseignement.
- Le message **persuasif**, visant à influencer.
- Le message **subjectif,** visant à diffuser une opinion.

Cependant, tous les Serious Games n'ont pas pour finalité de diffuser un message. Dans le corpus de jeux que nous avons étudiés, seuls 93% des titres visent une telle finalité *(p.32)*. Nous recensons donc des jeux appartenant aux catégories *« Training and Simulation Games »* ou *« Games for Health »* qui visent une autre finalité : **prodiguer un entraînement**. Par exemple, *Pulse!!* ***(Breakaway, 2007)*** sert à entraîner les médecins urgentistes à gérer des

situations de crise, tandis que *MoSBE* ***(Breakaway, 2007)*** permet de préparer des soldats à des opérations militaires. La notion d'entraînement se traduit ici par le développement de compétences physiques ou cognitives suite à la pratique du jeu. Néanmoins, il peut être considéré que tout jeu, y compris ceux destinés au divertissement, nécessite une certaine pratique pour être « gagné ». Pour déterminer le caractère « sérieux » d'un entraînement, nous pouvons nous référer à la définition des « Serious Games » *(p.18)*. Cette dernière stipule que leurs concepteurs les destinent à *« une finalité autre que le simple divertissement »*. En conséquence, si le concepteur a explicitement cherché à ce que l'entraînement prodigué par son jeu soit applicable en dehors du divertissement, alors ce jeu entre dans le champ du *« Serious Game »*. Si cet entraînement dépasse le divertissement par le biais d'un détournement vidéoludique qui n'a pas été anticipé par le concepteur du jeu, alors il rentre dans le champ du *« Serious Gaming »* *(p.22)*. Enfin, si cet entraînement ne dépasse pas le cadre du *« simple divertissement »*, alors nous restons dans le champ du « jeu vidéo » *(p.25)*.

Nous recensons donc, d'un côté, des jeux visant à diffuser un message, et de l'autre des titres destinés à prodiguer un entraînement. Pourtant, aucune des classifications existantes n'est en mesure de rendre compte de cette différence *(p.27)*. Cette dernière nous semble pourtant fondamentale dans le cadre d'une classification par « finalité ». Une troisième finalité, moins répandue, nous semble également intéressante à recenser pour classifier les Serious Games : les jeux destinés à **favoriser l'échange de données.** Dans ce registre, nous trouvons par exemple *Google Image Labeler* ***(Google, 2007)***. Ce Serious Game propose aux internautes de jouer en collaboration avec un partenaire choisi aléatoirement pour décrire une série de photographies. L'interface propose un champ de saisie où chaque joueur doit écrire une série de mots pour décrire au mieux l'image affichée. A la fin du temps écoulé, le jeu compte le nombre de mots en commun proposés par les deux joueurs pour leur attribuer des points de score. Une nouvelle photographie est alors présentée aux deux joueurs. Ce Serious Game a été développé par la société ***Google*** dans l'optique d'améliorer la pertinence de son moteur de recherche d'images. Chaque partie jouée est un moyen d'enrichir sa base de données de mots-clés à associer aux images. Ce type d'application, appelé *« datagame »*, littéralement « jeu sur les données », est encore assez peu répandu à ce jour. De tels *« datagames »* peuvent aussi contribuer à la recherche scientifique. Par exemple, *Foldit* ***(University of Washington, 2008)*** sollicite les internautes pour trouver des solutions innovantes en matière de pliage de protéines. Loin de se limiter à une collecte à sens unique, d'autres titres comme *PowerUP* ***(IBM, 2007)*** et *Lure of the Labyrinth* ***(The Education Arcade, 2009)*** sont explicitement destinés à favoriser les échanges entre joueurs, ici pour des informations à caractère éducatif.



7. Les Serious Games *Google Image Labeler* et *Foldit*

En résumé, nous proposons de classifier les finalités selon trois grandes catégories :

- **Diffuser un message:** le Serious Game vise à diffuser un ou plusieurs messages. Ces derniers peuvent être de quatre natures différentes : éducatif *(ex: Edugames)*, informatif *(ex: Newsgames)*, persuasif *(ex: Advergames)* et subjectif *(ex: Militant games, Art games)*. Un même jeu peut cumuler plusieurs natures de message.

- **Prodiguer un entraînement:** le Serious Game vise à améliorer les capacités cognitives ou physiques du joueur *(ex: Exergames)*.
- **Favoriser l'échange de données:** le Serious Game est destiné à favoriser l'échange de données *(ex: Datagames)* entre les joueurs, ou entre le diffuseur du jeu et les joueurs.

2.2.3 CRITERES DE CLASSIFICATION DE L'ASPECT « SECTEUR »

Ce critère propose deux niveaux d'informations complémentaires. Tout d'abord, une information sur le domaine d'application visé par le Serious Game. Cette liste de « marchés » doit régulièrement être mise à jour pour refléter l'apparition de nouveaux secteurs. Elle comporte, à ce jour, les domaines suivants : *Etat & Gouvernement, Militaire, Santé, Education, Entreprise, Religion, Art & Culture, Ecologie, Politique, Humanitaire & Caritatif, Médias, Publicité, Recherche Scientifique*.

L'autre information concerne le public visé qui est retranscrit par tranches d'âge ainsi que par trois types : *Grand Public, Professionnels, Etudiants*. Par exemple, pour le domaine de la *« Santé »*, les praticiens seront considérés comme *« Professionnels »*, les étudiants en médecine comme *« Etudiants »*, et les patients comme *« Grand Public »*. Cette information peut, bien entendu, être plus détaillée selon les besoins, en cherchant par exemple à identifier l'âge, le sexe, la nationalité… du public ciblé. Mais pour notre classification générale de l'ensemble des Serious Games, nous nous en tenons à ces trois « types » ainsi qu'à huit « tranches d'âge » différentes.

2.2.4 APPLICATION DE CE MODELE CLASSIFICATOIRE

En résumé, est considéré comme « Serious Game » tout jeu intentionnellement conçu afin de viser une ou plusieurs des *« finalités »* définies et dont le *« secteur »* n'est pas uniquement celui du *« Divertissement »*. Et ce, quelle que soit la nature de son *« gameplay »*. Nous avons alors utilisé ce système de classification pour tenter d'analyser un large corpus de Serious Games, tel que détaillé dans la section suivante.

3 UNE VUE D'ENSEMBLE DES SERIOUS GAMES

Afin d'obtenir une première « vue d'ensemble » des Serious Games, nous avons commencé la constitution d'un large corpus de titres pour effectuer des analyses quantitatives. Nous présenterons tout d'abord notre méthode de constitution de ce corpus, avant d'étudier quelques données extraites d'un ensemble de plus de 2000 Serious Games.

3.1 METHODOLOGIE

Lors du démarrage de notre travail de recherche, il n'existait pas véritablement de base de données recensant tous les Serious Games. Bien que de tels sites existent pour le jeu vidéo *(Mobygames, Allgames, GameFAQs…)*, aucun d'entre eux ne s'intéressait au secteur du Serious Game. En 2008, nous avons donc décidé de créer notre propre base de donnée en ligne, et de l'ouvrir selon les modalités du *« Web 2.0 »* aux contributions de tout internaute. Ainsi, le site Internet *Serious Game Classification*[13] vise à référencer le plus grand nombre possible de Serious Games, et à les classifier en utilisant le *modèle G/P/S* *(p.28)*. À ce jour, cette base de données recense **2218 Serious Games**.

[13] http://serious.gameclassification.com/

Cependant, nous avons vu que plusieurs approches existent pour définir ce qu'est un « Serious Game » *(p.18)*. Lors de la constitution de ce corpus nous nous sommes appuyé sur la définition d'un « Serious Game » proposée par ***Chen & Michael*** (2005) :

> *« Tout jeu dont la finalité première est autre que le simple divertissement »*[14].

Afin de limiter l'envergure du travail de recherche lié à la constitution du corpus, nous avons dans un premier temps recensé uniquement des jeux sur support informatique ou assimilé. Autrement dit, des « Serious Games » sous forme de jeu vidéo. Pour chaque jeu du corpus, plusieurs informations sont alors indexées dans la base de donnée : *titre, date de publication, noms et pays du développeur et de l'éditeur du jeu, support de jeu, mode de distribution…* Ces informations de base sont ensuite complétées par les différents critères du système de classification *G/P/S*. Pour déterminer si le jeu est un « Serious Game » ou bien un « jeu vidéo de divertissement », nous regardons tout simplement les critères *« Permet de »* et *« Secteur »*. Si le *« Secteur »* référencé est autre que *« Divertissement »* et qu'au moins une des finalités définie par *« Permet de »* a été attribuée au titre, alors il s'agit d'un « Serious Game ». Selon ce protocole, 405 contributeurs ont référencé quelques 15301 jeux vidéo différents sur le site *Game Classification*[15]. Sur ces 15301 jeux vidéo, 2218 titres remplissent les critères que nous venons de définir pour qualifier un « Serious Game ». Le site *Serious Game Classification* est donc une version « filtrée » de la base de données de *Game Classification*, qui n'affiche que les jeux qualifiés de « Serious Games ».

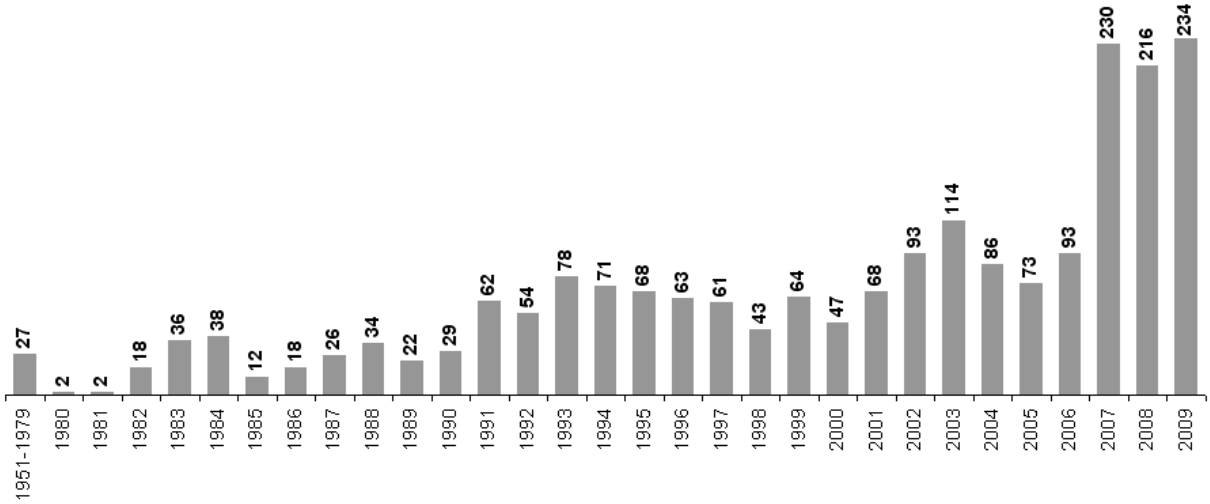
D'un point de vue méthodologique, rappelons que les données de notre corpus proviennent en partie de contributions extérieures. Par exemple, nous avons sollicité de nombreux studios de création de Serious Games pour qu'ils référencent leurs réalisations, et avons également reçu de nombreuses contributions de la part d'autres chercheurs du domaine. Bien que le chiffre de 2218 Serious Games soit relativement conséquent, il est très loin d'être exhaustif. De nombreux Serious Games manquent à ce corpus, ce qui introduit un certain biais dans l'analyse. De plus, il faut également prendre en compte le fait qu'une large part de nos contributeurs est francophone. Ainsi, les Serious Games publiés en France, Belgique ou Canada semblent mieux représentés dans notre corpus que les Serious Games publiés par exemple au Royaume-Uni ou en Corée. Les Serious Games américains bénéficiant généralement d'un rayonnement international, il ne semblent pas soumis à ce biais.

Pour résumer, les données quantitatives exposées dans ce chapitre ne sont donc pas à prendre pour un reflet exact et exhaustif des Serious Games existants dans le monde. Mais, après plus de trois ans passés à constituer ce corpus, il s'agit, à la date d'écriture de cette thèse, de la plus grande liste de « Serious Games » disponible. Faute de l'existence d'une base de données plus complète en la matière, nous estimons donc que ce corpus de Serious Games permet d'illustrer certaines tendances générales sur les différents titres composant ce domaine. De plus, malgré les potentiels biais liés à sa constitution, le chiffre de 2218 Serious Games nous semble être suffisamment représentatif[16]. Nous proposons donc d'analyser quelques données quantitatives extraites de ce corpus.

3.2 EVOLUTION DU NOMBRE DE SERIOUS GAMES DANS LE TEMPS

Commençons par étudier le nombre de Serious Games publiés chaque année, tels que synthétisé dans le schéma ci-dessous :

[14] *"Games that do not have entertainment, enjoyment or fun as their primary purpose"*
[15] http://www.gameclassification.com/
[16] A titre comparaison, le nombre total de jeux vidéo sortis sur *Wii* dans le monde est estimé à un millier.



8. Evolution du nombre de Serious Games publiés chaque année

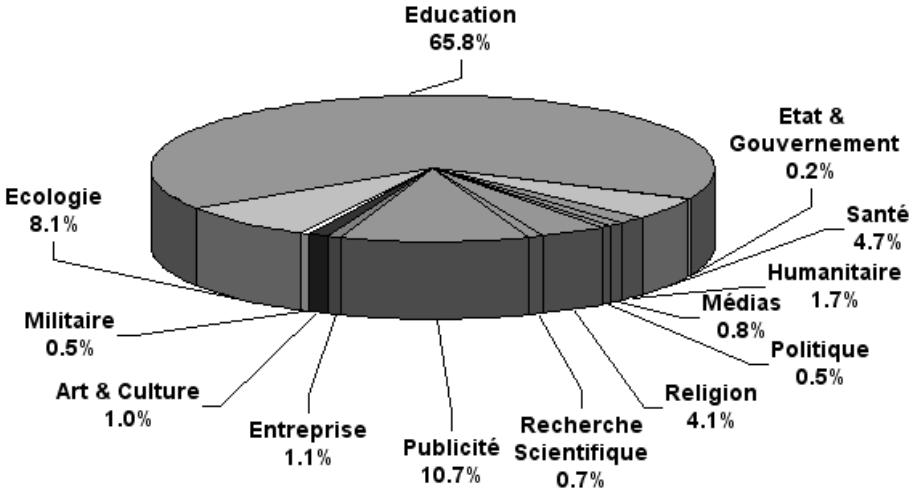
Nous constatons tout d'abord que le nombre de Serious Games publiés dans les années 2000 est plus important que les années précédentes, confirmant donc l'importance donnée à la « vague actuelle » des Serious Games *(p.18)*. Nous remarquons cependant que de nombreux jeux correspondants aux définitions actuelles du « Serious Game » ont été publiés avant l'année 2002. Ces « ancêtres » des Serious Games remontent même jusqu'au début de l'histoire du jeu vidéo. En effet, les premiers jeux sur support informatique, alors inventés par des chercheurs, n'étaient pas destinés au divertissement (J. Alvarez & Djaouti, 2010). Par exemple, *OXO* ***(Alexander Douglas, 1952)***, qui est pour l'instant considéré comme le premier jeu vidéo de l'histoire (Donovan, 2010), a été conçu pour illustrer un travail de recherche en informatique sur les interfaces homme-machine (Cohen, 2009). Dans un esprit similaire, *NIMROD* ***(John Makepeace Bennet, 1951)*** a été conçu pour promouvoir les ordinateurs du constructeur ***Ferranti*** (Smillie, 2010), tandis que *Tennis For Two* ***(William Higinbotham, 1958)*** tentait de rassurer les habitants d'une ville abritant un centre de recherche sur le nucléaire quant aux effets bénéfiques de ces travaux scientifiques (Anderson, 1983). Au-delà de ces jeux pionniers réservés aux laboratoires de recherche, la première console de salon de jeux vidéo, la *Magnavox Odyssey* ***(Ralph Baer, 1972)*** proposait à la fois des jeux de divertissement, tels que *Tennis*, *Haunted House* et *Roulette*, comme des jeux à caractère éducatif, à l'image d'*Analogic*, *States* et *Simon Says* (Baer, 2005).

Au fur et à mesure du développement de l'industrie du jeu vidéo en tant que secteur de divertissement, de nombreux autres titres ont suivi ces exemples pionniers. Commercialisés selon des modalités identiques aux titres de divertissement, ces jeux vidéo visent cependant des finalités qui s'en écartent. Par exemple, *The Oregon Trail* ***(MECC, 1971)*** est un jeu pour ordinateur destiné à enseigner la période historique de la conquête de l'ouest aux élèves américains. Ce célèbre titre éducatif connaîtra de nombreuses suites et adaptations au fil des ans. Dans un autre registre, *The Bradley Trainer* ***(Atari, 1981)*** est une version du populaire jeu d'arcade *Battlezone* ***(Atari, 1980)*** réalisée spécialement pour l'armée, à des fins d'entraînement interne. Disponible sur console de salon, *Kool Aid Man* ***(Mattel Electronics, 1983)*** est quant à lui destiné à faire la promotion des boissons de la marque *Kool Aid*. L'abus de sucre est justement une des préoccupations de *Captain Novolin* ***(Raya Systems, 1992)***, conçu pour apprendre aux joueurs comment gérer leur diabète au quotidien. De son côté, *Captain Bible in the dome of Darkness* ***(BridgeStone Multimedia Group, 1994)*** se consacre à l'enseignement de la religion catholique, tandis que *Versailles : complot à la cour du Roi Soleil* ***(Cryo, 1997)*** permet aux joueurs d'enrichir leur culture sur le célèbre palais royal. Et il ne s'agit là que de quelques exemples des nombreux jeux vidéo que nous avons recensé au sein de notre corpus. Face à cette multitude de « Serious Games » publiés avant 2002, nous pouvons légitimement nous interroger sur les différences entre les Serious Games de la vague actuelle et leurs ancêtres. Puisque tous ces jeux répondent aux même définitions quelle que

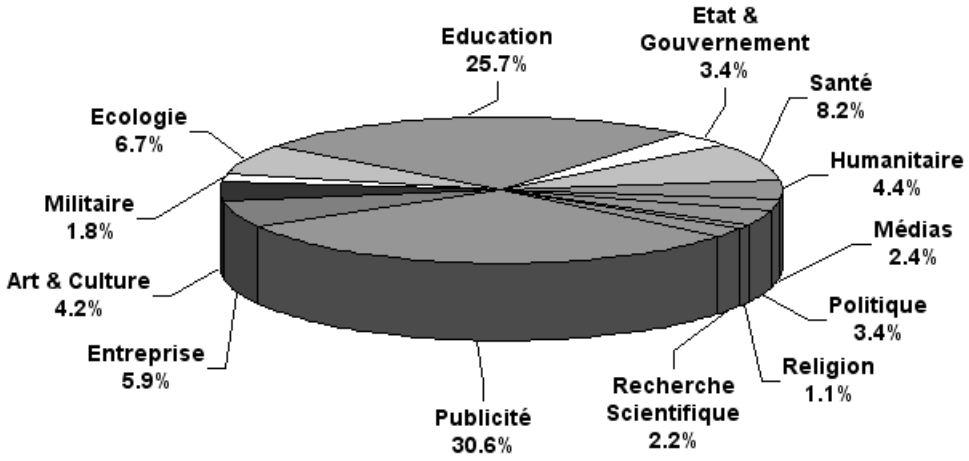
soit leur date de sortie, nous devons explorer d'autres données quantitatives pour tenter de les distinguer.

3.3 LES DIFFERENTS SECTEURS D'APPLICATIONS DES SERIOUS GAMES

Afin de tenter de différencier les Serious Games de la vague actuelle et leurs ancêtres, nous nous sommes intéressés aux secteurs d'applications de ces deux ensembles de jeux vidéo. Nous avons donc analysé séparément la répartition des « marchés » pour les Serious Games sortis entre 1951 et 2001 *(les « ancêtres », soit 953 jeux)* et ceux sortis depuis 2002 *(la « vague actuelle », soit 1265 jeux)*.



9. Répartition des marchés visés par les « Serious Games » sortis avant 2002 (953 titres)



10. Répartition des marchés visés par les « Serious Games » sortis à partir de 2002 (1265 titres)

Sur les 953 « ancêtres » des Serious Games de notre corpus, 65.8% d'entre eux ont été conçus pour le marché de l'éducation. Nous observons également 10.7% de jeux destinés à des fins publicitaires, et 8.1% qui traitent d'écologie. Les autres secteurs représentent chacun moins de 2% des titres. D'après ces données, les « ancêtres » des Serious Games sont majoritairement des jeux vidéo à caractère éducatif. La situation est différente du côté des jeux de la « vague actuelle », dans laquelle la part de l'éducation n'est plus que de 25.7%. En conséquence, les autres secteurs voient leur part augmenter, en particulier le secteur de la publicité qui représente dorénavant 30.6% des titres. Les autres marchés qui étaient pour la plupart sous la barre des 2%, se situent dorénavant dans une fourchette comprise entre 4% et 10% des jeux.

Au final, nous constatons donc que la plupart des ancêtres des « Serious Games » sont des « jeux éducatifs » contrairement aux titres de la vague actuelle. D'après ces données, nous pouvons donc penser que la « vague actuelle » permet aux Serious Games de s'inscrire dans une plus large variété de secteurs Nous remarquons également un plus grand nombre de titres présents au sein de la vague actuelle. En effet, les 953 ancêtres des Serious Games de notre corpus sont sortis entre 1951 et 2001 *(50 ans)*, alors que les 1265 Serious Games de la « vague actuelle » n'ont été publiés qu'entre 2002 et 2010 *(8 ans)*. Bien que les « Serious Games » et leurs ancêtres répondent aux mêmes définitions, nous voyons ici un moyen de les différencier. Les « Serious Games » de la vague actuelle ne se distinguent pas de leurs ancêtres en tant que jeux isolés, mais en tant qu'un ensemble plus volumineux de jeux vidéo qui traite d'une plus grande variété de sujets. Plusieurs facteurs permettent alors d'expliquer ces différences : *évolution de la place du jeu vidéo dans la société, changement de modèles économiques, vieillissement de la « génération Y »…* L'explication de ce phénomène se trouve être une thématique de recherche à part entière (J. Alvarez & Djaouti, 2010). Pour cette thèse, nous nous limiterons à cette observation des différents domaines d'applications visés par les « Serious Games ». Afin de compléter ces donnés quantitatives, découvrons à présent quelques exemples de Serious Games issus de la vague actuelle.

3.4 QUELQUES EXEMPLES DE SERIOUS GAMES ISSUS DE LA « VAGUE ACTUELLE »

Les données quantitatives sur les « marchés » des Serious Games montrent la diversité des thématiques qu'ils permettent d'aborder. Afin d'illustrer cette diversité, nous proposons ci-dessous quelques exemples de Serious Games de la « vague actuelle » issus de notre corpus. Chaque jeu est classifié selon les différents critères du *modèle G/P/S* *(p.28)*.

3.4.1 AMERICA'S ARMY (VIRTUAL HEROES, 2002)

Il s'agit d'un « jeu de tir en vue subjective » dans lequel le joueur incarne un soldat devant accomplir diverses missions au sein d'une escouade. Ce jeu multijoueurs est diffusé gratuitement par l'armée américaine afin d'aider au recrutement de nouveaux soldats. Il est également utilisé en interne comme outil d'entraînement.

- **Gameplay :**
  - Type : *Ludus.*
  - Objectifs: *Eviter, Atteindre, Détruire.*
  - Moyens: *Déplacer, Tirer, Gérer, Choisir.*
- **Permet de :**
  - *Diffuser un message persuasif*
  - *Prodiguer un entraînement.*
- **Secteur :**
  - Marchés: *Défense & Militaire, Etat & Gouvernement.*
  - Public visé: *à partir de 16 ans, Grand Public & Professionnels.*

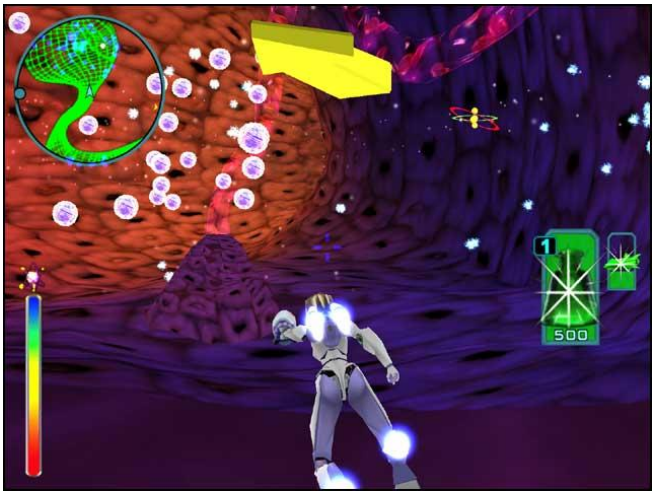
3.4.2 SEPTEMBER 12TH (GONZALO FRASCA, 2003)

Ce titre place le joueur face à un village du Moyen-Orient, peuplé d'habitants innocents et de « terroristes ». Le seul moyen d'action proposé au joueur est de tirer un missile, qui, après un certain temps de latence, ira exploser à l'endroit désigné. Si, ce faisant, le joueur tue une personne innocente, les villageois pacifiques assistant au massacre se transformeront alors en « terroristes ». La meilleure stratégie pour éviter la prolifération des terroristes au sein de ce jeu est donc tout simplement de ne jamais tirer… Diffusé gratuitement par son auteur, il s'agit d'un Serious Game destiné à faire réfléchir le grand public sur la nature de la réponse américaine aux attentats du 11 septembre.

- **Gameplay :**
  - Type : *Paidia.*
  - Objectifs: *(aucun).*
  - Moyens: *Tirer.*
- **Permet de :**
  - *Diffuser un message subjectif*
- **Secteur :**
  - Marchés: *Politique, Médias.*
  - Public visé: *à partir de 15 ans, Grand Public.*

3.4.3 RE-MISSION (HOPELAB, 2006)

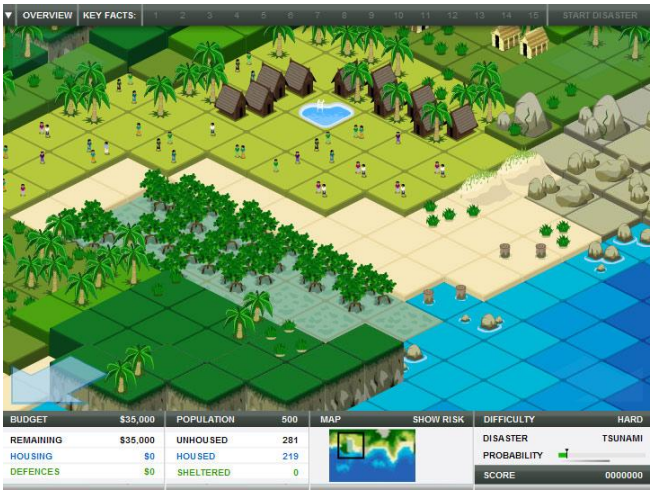
Ce Serious Game est destiné à des adolescents atteints de cancer. Il s'agit d'un jeu de tir en 3D temps réel se déroulant à l'intérieur du corps humain. Aux commandes d'une chimiothérapie personnifiée, la mission du joueur est d'éradiquer les différents types de cellules cancéreuses présentes dans le corps de plusieurs patients. Ce titre est utilisé dans le milieu hospitalier pour expliquer à de jeunes malades le fonctionnement de leurs traitements. Cela permet à l'équipe soignante d'amorcer un dialogue à propos de la maladie. Ce titre est également diffusé auprès du grand public afin de le sensibiliser à la lutte contre le cancer.



- **Gameplay :**
  - Type : *Ludus.*
  - Objectifs: *Eviter, Atteindre, Détruire.*
  - Moyens: *Déplacer, Tirer.*
- **Permet de :**
  - *Diffuser un message éducatif*
  - *Diffuser un message informatif*
- **Secteur :**
  - Marchés: *Santé.*
  - Public visé: *de 8 à 25 ans, Grand public.*

3.4.4 STOP DISASTERS! (PLAYERTHREE, 2007)

Ce Serious Game est un jeu de gestion dans lequel le joueur prend les commandes d'un village menacé par une catastrophe naturelle imminente *(tsunami, tremblement de terre…)*. L'objectif du joueur est d'arriver à aménager le village afin de limiter au maximum les victimes humaines, tout en garantissant un minimum de qualité de vie aux habitants *(soins, éducation…)*. Ce titre est diffusé gratuitement par l'***ONU*** afin de sensibiliser le grand public aux catastrophes naturelles, et de lui transmettre les rudiments de leur prévention.



- **Gameplay :**
  - Type : *Ludus.*
  - Objectifs: *Eviter, Atteindre.*
  - Moyens: *Créer, Gérer, Choisir.*
- **Permet de :**
  - *Diffuser un message éducatif*
  - *Diffuser un message informatif.*
- **Secteur :**
  - Marchés: *Humanitaire & Caritatif, Ecologie*
  - Public visé: *de 9 à 16 ans, Grand Public.*

3.4.5 LURE OF THE LABYRINTH (THE EDUCATION ARCADE, 2009)

Il s'agit d'un jeu d'aventure en ligne destiné à l'apprentissage des mathématiques. Au sein d'un univers fantastique, chaque joueur dirige un avatar. Il peut alors s'adonner à un ensemble de petits jeux simples lui demandant de mobiliser les fondamentaux mathématiques. La grande force de ce titre est sa composante multijoueurs, qui vise à renforcer l'échange d'informations à caractère éducatif entre les élèves. Pour cela, les joueurs gagnent des points de score s'ils rédigent des aides à même d'aider leur camarades à résoudre les énigmes du jeu. Ce Serious Game est conçu pour être utilisé en classe, chaque enseignant ayant la possibilité de préparer une session de jeu « sur mesure » pour ses élèves. De plus, le site officiel du jeu propose des fiches pédagogiques à destination des enseignants et des parents.

- **Gameplay :**
  - Type : *Ludus.*
  - Objectifs: *Eviter, Atteindre.*
  - Moyens: *Créer, Gérer, Déplacer, Choisir, Ecrire.*
- **Permet de :**
  - *Diffuser un message éducatif*
  - *Favoriser l'échange de données*
- **Secteur :**
  - Marchés: *Education.*
  - Public visé: *de 11 à 15 ans, Etudiants.*

4 QUELQUES TENDANCES ACTUELLES DU SECTEUR DU SERIOUS GAME

Le secteur du Serious Game, qui regroupe un nombre croissant d'acteurs d'origines variées, est en constante évolution. De nombreuses « tendances » sont mises en évidences par les spécialistes du secteur quant à ses possibles évolutions. Il se trouve que trois de ces « tendances » sont justement en lien avec la question de la création de Serious Games, car elles visent à essayer de la faciliter.

4.1 REALISER DES ECONOMIES D'ECHELLE AU SEIN DE L'INDUSTRIE

Dans un article de presse, ***Stéphane de Buttet*** (2010) expose les différents modèles économiques de la vague actuelle des Serious Games. Il note principalement la montée en puissance du modèle économique dit « à la licence ». Au lieu de se voir proposer la création d'un Serious Game « sur mesure », un commanditaire potentiel peut se voir offrir la personnalisation de Serious Games plus « génériques ». Si le projet final est forcément moins spécifique aux besoins du client, le coût total du projet s'en trouve amoindri, le temps de réalisation du jeu étant plus court. Du point de vue des studios de développement, la création de tels Serious Games « génériques » passe par la mutualisation et l'industrialisation d'outils de conception et de fabrication idoines (Boudier & Dambach, 2010).

Ainsi, nous voyons apparaître des « frameworks » ou autres « usines logicielles » qui permettent de créer des « briques logicielles réutilisables » de manière à accélérer le développement industriel de Serious Game. Au niveau national, de tels outils sont actuellement inventés en partenariat avec des laboratoires de recherches. Ainsi, le framework *SeGAE*, utilisé par la société ***KTM Advance***, est le fruit d'une équipe de chercheurs du ***LIP6***

(Yessad, Labat, & Kermorvant, 2010). Sur un moteur de jeu de gestion réalisé par des programmeurs expérimentés, cet outil propose un éditeur visuel permettant de créer simplement des événements logiques. Un concepteur sans connaissances en programmation peut donc créer ses propres scénarios de manière relativement poussée en sélectionnant des « conditions » et des « actions » à travers des menus déroulants. Suivant une logique similaire, le projet *Learning Game Factory* réunit ***Symetrix, SBT***, ***Daesign***, ***Les Tanukis*** et ***Genezis*** pour le côté industriel, ainsi que ***l'équipe de recherche de l'ESC Chambéry***, le ***Laboratoire informatique de Grenoble (INPG-LIG)***, l'***INSA Lyon*** et ***l'équipe SysCom de l'Université de Savoie*** pour les chercheurs (Deguerry, 2009). Ce projet vise à proposer :

> *« [U]ne plateforme d'édition et de diffusion web de learning games pour les grands comptes, les PME, les écoles et le grand public. En vue, tout simplement l'industrialisation des Serious Games avec la possibilité de créer plus vite et moins cher des learning games ! »*

Pour cela, l'outil imaginé par les partenaires met à disposition un ensemble de « modules réutilisables », qui seront développés par des programmeurs expérimentés. Des concepteurs pourront ensuite piocher dans cette base de modules pour construire leur propre Serious Game, réduisant ainsi considérablement le temps de fabrication *(p.91)*.

Les projets d'outils permettant d'accélérer la production industrielle de Serious Game ne sont pas forcément destinés uniquement aux concepteurs professionnels de jeux. Par exemple, le projet de *Serious Game Journalistique*, monté par le journal ***Le Monde***, ***KTM Advance***, et l'***Ecole Supérieure de Journalisme de Lille,*** vise à proposer des outils permettant à des journalistes de produire rapidement des Serious Games traitant de sujets d'actualité (Vandamme, 2010). Dans une veine similaire, le projet *Moteur de Jeux Orientés Santé (MoJOS)* rassemble la société ***DIDACT Systèmes***, le cabinet d'étude ***IDATE***, et deux laboratoires de l'***Université de Montpellier*** spécialisés dans la recherche médicale ***(UM1-CHU)*** et informatique ***(UM2-LIRMM)***. Cet outil vise à accélérer la production de Serious Games thérapeutiques, qui seront par exemple utilisés pour la rééducation de patients victimes d'un accident vasculaire cérébral (Harmonie Magazine, 2009). Dans un registre similaire, un des projets de recherche de la ***North Carolina State University*** a abouti à la création d'un outil baptisé *Virtuoso*. Sa vocation est de permettre à des enseignants de créer des scénarios pédagogiques pour des jeux d'action-aventure en 3D *(p.220)*. En parallèle à ces partenariats entreprise-université, certaines sociétés spécialisées dans la création de Serious Games commercialisent également leurs propres outils de travail, à l'image de la société ***Caspian Learning*** et de son outil *Thinking Worlds* *(p.214)*. Plus récemment, le studio français ***Onlineformapro*** propose un outil baptisé *SGTools* (Onlineformapro, 2010), dont le fonctionnement est largement inspiré par la famille de logiciels *Virtools* *(p.182)*.

4.2 PERSONNALISER DES SERIOUS GAMES POUR AIDER LE « SERIOUS GAMING »

La montée en puissance de la pratique du *« Serious Gaming »* *(p.22)* amène de nouvelles problématiques. De plus en plus encouragée dans le secteur de l'éducation (Gee, 2007; Prensky, 2007; Shaffer, 2007), cette pratique se voit même formalisée à travers des « guides » proposant un choix de « jeux vidéo à détourner pour l'éducation ». Si la plupart sont sous la forme de sites Internet, tel le blog tenu par ***Yasmine Kasbi***[17], il en existe aussi au format papier, comme le *catalogue ENGAGE Learning 2009-2010*, réalisé grâce à un financement de la communauté européenne (Pivec, 2010).

[17] http://yasminejoue.wordpress.com/

Pour autant, il peut arriver qu'un enseignant qui souhaite détourner des jeux vidéo pour les utiliser en classe soit confronté à des aspects du jeu qui ne conviennent pas à sa méthode pédagogique. Qu'il s'agisse d'un thème mal abordé, d'un jeu trop long, ou tout simplement d'informations incomplètes, un enseignant peut rapidement sentir le besoin de « modifier » quelque peu le contenu du jeu pour l'adapter à son cours. Face à ce problème, une équipe de chercheurs canadiens a monté un projet visant à proposer des *Coquilles Génériques de Jeux Educatifs*. Il s'agit de jeux pour lesquels chaque enseignant peut créer son propre contenu, et ainsi adapter au mieux le jeu qu'il souhaite utiliser en classe *(p.227)*. Si ce type de projet vise à concevoir des « jeux modifiables » spécifiquement destinés à un public de pédagogues, d'autres enseignants font tout simplement le choix d'utiliser des jeux vidéo du commerce dotés en standard de fonctionnalités de personnalisation. Par exemple, le collectif ***Pedagame*** a utilisé le jeu *Buzz! Quiz TV* ***(Relentless Software, 2008)*** car chaque enseignant pouvait intégrer les questions qu'il souhaitait au jeu avant de le présenter à ses élèves *(p.23)*.

Ainsi, les possibilités de « personnalisation » d'un jeu commencent à rentrer en compte lors de son détournement dans le cadre du *« Serious Gaming »*. Dans une certaine mesure, cette volonté peut être mise en parallèle avec le fait que certaines sociétés de l'industrie du Serious Game souhaitent offrir à leurs clients la possibilité de créer leurs propres scénarios. C'est par exemple le cas de la société ***OKTAL***, spécialisée dans les simulateurs de formation à la conduite de véhicules. Son logiciel *SCANeR* propose en standard une interface de « création de scénario », permettant à chaque client d'adapter ou de créer de nouvelles situations par lui-même, sans avoir à solliciter une prestation supplémentaire auprès de la société. ***OKTAL*** étant un des partenaires du *CIFRE* ayant permis le travail de recherche présenté dans cette thèse, nous reviendrons ultérieurement sur cet outil *(p.215)*.

4.3 DEMOCRATISER LA CREATION DE SERIOUS GAMES

A mi-chemin entre ces deux tendances, nous recensons également plusieurs projets visant à créer des outils permettant à des personnes ne possédant aucune connaissance en programmation ou en conception vidéoludique de réaliser ses propres jeux éducatifs. Les exemples que nous avons pu recenser à ce jour sont principalement des initiatives d'origines universitaires. Nous avons déjà mentionné *Virtuoso* *(p.220)* et les *Coquilles Génériques de Jeux Educatifs* *(p.227)*. ***Rankin & al.*** (2009) proposent quant à eux de s'appuyer sur un jeu de tir pour créer des jeux éducatifs. A partir d'*Unreal Tournament 2004* ***(Epic Games, 2004)***, les chercheurs ont tout d'abord construit de nombreux niveaux permettant d'aborder des concepts de chimie au niveau collège. La suite de leur travail, toujours en cours lors de la rédaction de cette thèse, vise à créer des outils permettant la réalisation de tels *« mods »* par des enseignants sans expérience préalable en conception et développement de jeux vidéo. Mais de tels projets visent parfois à permettre non pas aux enseignants, mais aux élèves, de créer leurs propres Serious Games. Ainsi, *Adventure Author* et *Flip* proposent des outils permettant simplifier la création vidéoludique pour la rendre accessible à un public jeune et/ou inexpérimenté en matière de programmation informatique *(p.225)*. Si tous ces projets s'appuient sur des outils et moteurs de jeux vidéo déjà connus et reconnus dans la sphère du divertissement, d'autres chercheurs ont choisi de créer de nouveaux logiciels permettant à des enseignants de créer des jeux éducatifs, à l'image de *<e-Adventure>* *(p.222)*.

INTRODUCTION AU GAME DESIGN

Comme nous venons de l'évoquer, le secteur du Serious Game est actuellement en phase de réflexion sur les différentes approches permettant de faciliter la création de tels jeux *(p.39)*. Plus précisément, ces réflexions visent tout d'abord à proposer aux professionnels de l'industrie de gagner en temps de production. Elles visent également à rendre la création de Serious Games plus accessible aux personnes extérieures à l'industrie, tels que les enseignants, afin de leur permettre de réaliser leurs propres jeux.

Le champ du jeu de divertissement a déjà abordé ces questions en essayant de réfléchir à des moyens permettant de mettre sa propre pratique créative à la portée d'amateurs *(p.14)*. Ce chapitre propose donc un regard sur le *« Game Design »*, afin d'esquisser un rapide état de l'art des pistes explorées pour faciliter la création de jeux vidéo de divertissement. La suite de notre travail de recherche vise à identifier d'éventuelles idées, concepts ou outils provenant du « Game Design » qui peuvent être bénéfiques au champ du « Serious Game ». Précisons que, bien que le « Game Design » recouvre a priori tout type de jeu, **nous limiterons notre analyse à la sphère des jeux vidéo**. Au delà de notre champ disciplinaire de rattachement, ce choix est avant tout motivé par le fait que la vague actuelle de « Serious Games » est principalement située sur support informatique, tel que nous venons de l'observer *(p.18)*.

1 UNE DEFINITION DU « GAME DESIGN »

Littéralement, le terme *« Game Design »* peut se traduire par « conception de jeu ». Cette traduction renvoie à la définition donnée par ***Salen & Zimmerman*** (2003). Les deux chercheurs définissent le « Game Design » comme :

> *« Le processus par lequel un concepteur crée un jeu, destiné à être utilisé par un joueur, afin que naisse une expérience de jeu »*[18] *[p.80]*

Cette définition reste vague quant à la nature exacte de ce « processus » de Game Design. Nous pouvons penser que cela est dû à l'apparente diversité des pratiques se rapportant à la création de jeux. Nous détaillerons ultérieurement ces différentes approches culturelles du « Game Design » *(p.46)*, car cette définition nous renvoie tout d'abord à une question fondamentale. **Si le « Game Design » est le processus de création d'un jeu, qu'est-ce que sous-tend exactement la notion de « jeu » ?**

2 UNE DEFINITION DU JEU

Cette vaste question a engendrée une littérature plus que conséquente, sans pour autant apporter une réponse ayant véritablement fait consensus (Caillois, 1967; Crawford, 2004; Esposito, 2005; Huizinga, 1951; Salen & Zimmerman, 2003, 2005). Face à ce constat, nous pourrions supposer qu'il existe non pas une, mais plusieurs définitions du « jeu », dont la pertinence varieraient avec l'angle d'étude choisi.

[18] "*Game design is the process by which a game designer creates a game, to be encountered by a player, from which meaningful play emerges"*

Une autre réaction face à cette multitude de définitions serait de les analyser de manière à identifier d'éventuels éléments récurrents permettant de proposer une définition synthétique. C'est justement le travail de recherche inauguré par ***Salen & Zimmerman*** (2003). Il fut repris et complété par ***Juul*** (2003), qui a analysé un total de sept définitions antérieures. Pour cette analyse, ***Juul*** pose comme postulat de départ qu'une bonne définition du jeu doit décrire trois aspects :

- Le jeu en tant qu'objet créé par un concepteur *(« the game »)*.
- La relation entre le joueur et le jeu *(« the player »)*.
- Le rapport du jeu au reste du monde *(« the world »)*.

Ces trois aspects représentent les trois « angles définitoires » proposés par ***Juul***. Ensuite, le chercheur, compare les définitions données par ***Huinzinga*** (1951), ***Caillois*** (2005), ***Suits*** (2005), ***Avedon & Sutton-Smith*** (1971), ***Crawford*** (1982), ***Kelley*** (1988), ainsi que ***Salen & Zimmerman*** (2003). A noter que cette dernière définition est elle-même le fruit de l'analyse synthétique de huit définitions issues des travaux de ***Parlett*** (1999), ***Abt*** (1970), ***Huinzinga*** (1951), ***Caillois*** (2005), ***Suits*** (2005), ***Crawford*** (1982), ***Costikyan*** (2005) et ***Avedon & Sutton-Smith*** (1971).

Juul identifie alors des ressemblances entre ces multiples définitions. Ces ressemblances lui permettent de proposer six « points-clés » définissant un jeu : les *« règles »*, le *« résultat quantifiable variable »*, la *« valorisation du résultat »*, *« l'effort du joueur »*, *« l'attachement du joueur au résultat »* et les *« conséquences négociables »*. A partir de ces six « points-clés », ***Juul*** propose la définition du « jeu » suivante :

> *« Un jeu est un système formel basé sur des règles donnant lieu à un résultat quantifiable variable. Ces variations de résultat sont associées à des valeurs différentes. Le joueur est donc émotionnellement attaché au résultat du jeu, et s'implique de manière à influencer le résultat produit. Cependant, les conséquences réelles d'une telle activité restent facultatives et négociables. »*[19]

Cette définition est associée à un tableau mettant en correspondances les six « points-clés » avec les trois angles définitoires de son postulat de départ :

Tableau 1. Tableau définitoire du « modèle classique du jeu » de *Juul*

	« The Game » *Le jeu en tant qu'objet créé par un concepteur*	**« The Player »** *La relation entre le joueur et le jeu*	**« The World »** *Le rapport du jeu au reste du monde*
1. Règles	**X**		
2. Résultat quantifiable variable	**X**		
3. Valorisation du résultat		**X**	
4. Effort du joueur	**X**	**X**	
5. Attachement du joueur au résultat		**X**	
6. Conséquences négociables			**X**

Ces six points-clés et leurs relations avec les trois angles définitoires constituent le *modèle classique du jeu* proposé par ***Juul***. Il précise que cette définition est transversale, au sens

[19] *"A game is a rule-based formal system with a variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort in order to influence the outcome, the player feels attached to the outcome, and the consequences of the activity are optional and negotiable."*

qu'elle concerne tous les types de jeux, quel que soit leur support matériel. Car en effet, un « jeu » peut s'appuyer sur plusieurs types de « supports » : cartes *(jeu de cartes)*, plateau *(jeu de société)*, ordinateur *(jeu vidéo)*, lois de la physique *(sport)*…

2.2 « JEU » ET « SUPPORT DE JEU »

Toujours selon ***Juul*** (2005), il n'existe aucun « support » particulier qui définisse le « jeu ». Le jeu peut donc s'appuyer sur une large gamme de « supports », à condition que ledit support possède deux caractéristiques :

- Une capacité de calcul permettant d'appliquer les règles du jeu en réponse aux actions du joueur.
- Une capacité à retenir l'état actuel du jeu, en mémorisant l'état de chacun des éléments qui composent le système de jeu.

Dans le cas des jeux de cartes, les cartes servent à retenir l'état actuel du jeu pendant que le cerveau humain se charge d'appliquer les règles. Dans le cas des sports, les lois de la physique et le cerveau de l'arbitre permettent d'appliquer les règles, tandis que des objets comme les compteurs de score ou les joueurs eux-mêmes permettent de mémoriser l'état courant du jeu. Pour le cas du jeu vidéo, le processeur de l'ordinateur *(CPU)* permet d'appliquer les règles tandis que la mémoire vive *(RAM)* de cette même machine permet de conserver l'état actuel du jeu.

A partir des réflexions de ***Juul***, nous pouvons donc différencier le « jeu » de son « support ». Cette distinction est parfaitement illustrée par les premiers jeux vidéo *(p.33)*, qui étaient des adaptations de jeux de plateau au support informatique (Djaouti, 2010a) :

- Les *échecs* se trouvent adaptés sur le *Manchester Mark I* par ***Prinz*** en 1951.
- Le *jeu de Nim* arrive sur ordinateur avec le *NIMROD*, conçu par ***Bennet*** en 1951.
- Le *morpion* est adapté sur *EDSAC* par ***Douglas*** en 1952 à travers son jeu ***OXO***.
- Les *dames* migrent sur support informatique à travers deux programmes de ***Strachey***, tout d'abord en 1951 pour le *Pilot Ace* puis en 1952 pour le *Manchester Mark I*.

Lors de la transposition de ces jeux du support « plateau » au support « informatique », les règles du jeu ont été conservées. Seul le « support », et donc la façon dont les règles sont appliquées, a changé. Bien que la mémoire et le processeur de l'ordinateur remplacent le plateau de jeu et le cerveau humain, les règles de jeu restent les mêmes. En nous plaçant du point de vue d'un créateur de jeu, nous pouvons alors identifier deux étapes distinctes :

- La conception des règles de jeu.
- La fabrication d'un support de jeu.

2.3 « GAME DESIGN » ET « GAME DEVELOPMENT »

Dans l'industrie du jeu vidéo de divertissement, l'étape de conception des règles de jeu et celle de la fabrication du support de jeu sont deux métiers distincts. Le processus visant à inventer des règles de jeu est baptisé **« Game Design »**. La finalité de cette étape est la rédaction d'une sorte de cahier des charges du nom de *« Game Design Document »* (Rogers, 2010; Rouse, 2001). Ce document détaille de manière très précise le fonctionnement d'un jeu vidéo afin de permettre sa fabrication. Support informatique oblige, la tâche de fabrication d'un jeu vidéo incombe à des développeurs, et est donc baptisée **« Game Development »**. Même si la séparation entre les métiers de concepteur et de fabricant est parfois moins évidente pour les jeux s'appuyant sur d'autres supports, nous la retrouvons également dans l'industrie du jeu de société (Tinsman, 2008). En effet, bien que le concepteur imagine en général les règles du jeu ainsi que le support matériel, lors de la réalisation industrielle d'un

jeu de société l'éditeur recommence l'étape de fabrication de son côté. Cela permet d'obtenir un jeu qui soit fabricable en usine, là où un concepteur fabriquera son propre jeu de plateau de manière artisanale. Pour un concepteur de jeu de société, la création du support de jeu se borne donc à la réalisation de « prototypes », en général avec les moyens du bord. Cette remarque est également valable dans l'industrie du jeu vidéo, où le concepteur est parfois amené à réaliser des prototypes sur support informatique. Cependant le code informatique de ces derniers ne sera pas utilisé par les développeurs chargés de la véritable fabrication du jeu vidéo, qui s'appuieront uniquement sur le *« Game Design Document »* (Fullerton, 2008).

Si la question du choix du support de jeu, voire de la définition de ses caractéristiques, relève bien du travail du *« Game Designer »*, la tâche de fabrication du jeu à proprement parler dépasse son champ de compétences. La fabrication d'un jeu, que ce soit sur support informatique ou tout autre support, requiert des compétences spécifiques *(maîtrise d'un langage de programmation informatique, utilisation d'outils pour découper du bois…)* qui ne relèvent pas directement de la conception de jeu, et donc du *« Game Design »*. Pourtant, nombreux sont les inventeurs de jeu, quel que soit le support visé, à posséder à la fois des compétences pour la conception et la fabrication d'un jeu. En effet, si la fabrication industrielle n'est pas la responsabilité du concepteur, nous avons vu que ce dernier fabrique généralement des prototypes par lui-même. Il a pour cela recours à des outils adaptés à son niveau de compétence technique, qui permettent de « fabriquer un jeu », même si c'est de manière artisanale et non industrielle. Pour revenir à l'exemple de l'industrie du jeu vidéo, si la fabrication d'un jeu se fera en programmant directement un moteur de jeu avec un langage commun tel que le *C++*, les prototypes pourront tout à fait être réalisés avec des logiciels auteurs plus simples d'accès, à l'image de *Flash* ***(Macromedia, 1996-2011)***.

Malgré la distinction que nous venons de faire entre les étapes de « conception » et de « fabrication » d'un jeu, ces deux étapes sont nécessaires pour créer un jeu. Notre problématique s'attachant à la facilitation de la création d'un Serious Game, elle vise potentiellement des approches ayant traits à la « conception » comme à la « fabrication » d'un jeu vidéo. La phase de « conception » s'appuie avant tout sur un travail de réflexion théorique, tandis que la phase de « fabrication » mobilise des nombreuses compétences techniques. Dans cette thèse, nous aborderons donc à la fois des **considérations théoriques** *(p.58)* et des **considérations techniques** *(p.155)* sur la création de Serious Games, afin d'identifier des approches permettant de la faciliter.

2.4 « GAME DESIGN » ET « LEVEL DESIGN »

L'industrie du jeu vidéo opère une autre différence entre deux notions liées à la création vidéoludique. Si nous venons d'aborder le fait qu'elle emploie la notion de « Game Design » pour renvoyer à la conception d'un jeu plutôt qu'à sa fabrication, il lui arrive également de réserver l'expression « Game Design » à une partie seulement de la conception. En effet, dans l'industrie, le travail d'un *« Game Designer »* se focalise généralement sur l'invention des règles et de l'univers du jeu. Une fois ces bases posées, plusieurs *« Level Designer »* se consacreront à la création des différents « niveaux » du jeu. Ces niveaux peuvent être vus comme autant de scénarios mettant en scène les mécanismes de jeu. Le travail du *« Level Designer »* s'appuie sur un outil logiciel spécifique, appelé *« éditeur de niveau » (p.50)*. Si nous avons défini le *« Game Design »* comme le processus de création d'un jeu vidéo, ce terme est également employé pour en désigner une partie spécifique. L'expression de **« Level Design »** est alors utilisée en complément, et renvoie à une autre partie de la conception d'un jeu vidéo. Cette notion de *« Level Design »* est d'ailleurs suffisamment caractérisée dans l'industrie pour bénéficier d'une littérature spécifique (Kremers, 2009).

Au final, nous identifions donc trois significations possibles pour le terme *« Game Design »* :

- Le processus global de création d'un jeu
- La phase de « conception » d'un jeu, distincte de celle de « fabrication »
- Une partie de la phase de « conception » : la création des mécanismes et de l'univers de jeu. Elle est complétée par une autre partie : le *« Level Design »*, qui renvoie à la construction des niveaux du jeu.

Malgré la confusion qu'elle peut engendrer, il est important de souligner **cette pluralité de significations du terme « Game Design »**. Pour plus de clarté, dans cette thèse nous emploierons l'expression *« Game Design »* pour renvoyer à la globalité du processus de création d'un jeu.

3 DIFFERENTES APPROCHES DE FACILITATION DU GAME DESIGN

Au-delà sa pluralité de définitions, le « Game Design » renvoie également à une large variété de pratiques dans l'univers du jeu vidéo. Comme nous venons de l'évoquer, le Game Design se défini comme un processus aboutissant à la création d'un jeu. Mais ce processus peut varier selon le cadre dans lequel il est pratiqué. Par exemple, il existe de nombreuses différences entre un Game Designer professionnel oeuvrant sur un titre *« AAA »*[20] et un Game Designer amateur cherchant à créer une variante de son jeu favori. Ces différences portent notamment sur les **outils** utilisés et le **niveau de formation** théorique et technique des concepteurs. Mais surtout, elles portent sur la **finalité** de l'acte créatif : là où un professionnel sera rémunéré pour produire un jeu répondant à un cahier des charges, un amateur créera durant son temps libre et pour son seul plaisir.

En prenant en compte ces différences, nous observons plusieurs types d'approches visant à simplifier le Game Design. Plus précisément, nous identifions trois « cultures » du Game Design de jeu vidéo, qui correspondent à des profils de créateurs distincts. Ces trois « cultures » proposent des approches de facilitation différentes. Afin de ne pas alourdir cette section, les aspects historiques relatifs à ces « cultures » sont regroupés en annexe *(p.307)*.

3.1 LES CREATEURS PROFESSIONNELS

Deux types de créateurs de jeux vidéo pratiquent le « Game Design » de manière professionnelle : les concepteurs de l'industrie et les créateurs indépendants.

3.1.1 LES GAME DESIGNERS DE L'INDUSTRIE DU JEU VIDEO

L'industrie vidéoludique produit de nombreux titres qui sont commercialisé dans les rayons « jeu vidéo » des magasins. La conception de ces jeux vidéo s'insère dans un processus de production industrielle (Rouet, 2009). En écho à la croissance économique permanente du secteur des jeux vidéo de divertissement, les budgets de productions sont de plus en plus importants (Natkin, 2006). En 2010, le budget moyen de création d'un jeu vidéo *« AAA »* se situe entre 18 et 28 millions de dollars (Meloni, 2010). A ce montant s'ajoutent les coûts de promotion et distribution du jeu, qui correspondent en général au double du budget de développement. L'industrialisation du processus de création de tels jeux vidéo s'accompagne donc de contraintes de rentabilité économique très fortes.

Une des approches permettant d'améliorer cette rentabilité est le recours au *« middleware »*. Les « middlewares » sont des outils logiciels permettant de créer des jeux vidéo plus

[20] « AAA » est terme qualifiant empiriquement les plus grosses productions de l'industrie du jeu vidéo de divertissement. Il s'agit de l'équivalent vidéoludique des « blockbusters Hollywoodiens ».

rapidement, et donc à moindre coût. Pour cela, ils proposent des « parties logicielles » déjà réalisées, évitant aux créateurs de jeux d'avoir à les réaliser par eux-mêmes. Ces « parties logicielles » touchent à tout ce qui constitue un jeu vidéo : *rendu graphique, gestion de la physique, règles de jeu…* Au départ, chaque studio réalisait ses propres « parties logicielles » et les conservait en interne pour les réutiliser d'un projet à l'autre. Aujourd'hui, la plupart de ces « parties logicielles » sont réalisées par des sociétés spécialisées, qui les commercialisent en tant qu'outils propres. Ce sont ces outils que l'on désigne par le terme *« middleware »* (De Prato, Feijoo, Nepelski, Bogdanowicz, & Simon, 2010). Parmi les « middlewares » se trouvent tous les « moteurs de jeu », comme *Unreal Engine 3* ***(Epic Games, 2007)***. Ce dernier permet de créer facilement des jeux de tir en vue subjective, ou tout autre jeu d'action en 3D, avec une qualité de rendu graphique très élevée. Mais les « middlewares » sont parfois plus spécialisés, l'image de *Havok* ***(Havok, 2000)***, un outil dédié uniquement à la gestion de la physique. Les spécialistes de l'industrie du jeu vidéo parlent alors de *« Middleware 2.0 »* (Develop, 2007). Cette appellation désigne les « middlewares » composés d'un ensemble de petits composants spécialisés dans une tâche précise, comme *Havok*. Ils s'opposent aux *« Middleware 1.0 »* qui proposent des solutions « tout-en-un », comme *Unreal Engine 3*. A noter que ces outils étant destinés à la fabrication de jeu vidéo, les approches de facilitation qu'ils mobilisent se focalisent plutôt sur la phase du *« Game Development »* que du *« Game Design »* *(p.44)*. Mais un concepteur de l'industrie se doit de prendre en compte les moyens techniques de réalisation d'un jeu vidéo lorsqu'il en invente le concept. Il doit donc posséder une certaine culture de ces divers outils de type *« middleware »*.

Les approches de facilitation dans l'industrie du jeu vidéo de divertissement ne se limitent pas à des outils techniques. L'industrie dispose également de méthodologies théoriques permettant de « formaliser » le processus de conception d'un jeu vidéo. Elle propose également des outils destinés à accompagner la réflexion créative, comme des modes de schématisation de la structure d'un jeu ou des catégorisations des divers profils de joueurs. Cette théorisation du Game Design pour l'industrie est généralement traitée par le biais d'ouvrages spécialisés, à l'image de ceux qui nous étudierons dans le chapitre suivant *(p.59)*. L'apparition et l'évolution constante d'outils théoriques et techniques pour le Game Design industriel augmente perpétuellement la quantité de savoirs et savoir-faire qui y sont associés. Si lors des débuts de l'industrie du jeu vidéo, la plupart des Game Designers étaient des autodidactes, ils sont aujourd'hui de plus en plus formés par le biais de cursus spécialisés. Une étude indique de 74% des professionnels aujourd'hui en poste dans l'industrie vidéoludique française sont issus de formations spécialisées (SNJV, 2009).

En résumé, les approches de facilitation du Game Design dans l'industrie s'appuient sur des outils théoriques *(méthodologies de conception, outils de réflexion)* et techniques *(middleware)*. Ces outils sont néanmoins suffisamment complexes pour que les professionnels reçoivent une formation idoine afin de les utiliser. Cette formation est dispensée au sein de cursus spécialisés ou à travers la lecture d'ouvrages. Conséquence du haut niveau de spécialisation du Game Designer évoluant dans l'industrie du jeu vidéo, ses tâches se limitent uniquement à quelques phases du processus de création d'un jeu vidéo. Sa principale mission est de produire le *« Game Design Document »* *(p.44)*. Ce document servira de guide aux nombreux autres corps de métiers intervenant dans la création industrielle d'un jeu vidéo *: graphistes, programmeurs, level designers, producteurs, testeurs…*

3.1.2 LES GAME DESIGNERS PROFESSIONNELS INDEPENDANTS

Tous les Game Designers professionnels ne s'inscrivent pas forcément dans la « grande industrie ». Il existe aujourd'hui de nombreux studios de développement de taille modeste *(de une à cinq personnes)* qui réalisent des jeux commercialisés par le biais de l'autoédition. Ils s'appuient pour cela sur Internet et les plateformes de distribution dématérialisées *(Steam, XBox Live, PSNetwork, Wiiware…)* pour diffuser leurs réalisations. S'ils partagent parfois la

formation et le savoir théorique des Game Designers de l'industrie, leurs budgets et temps de production sont nettement plus modestes. Ainsi, quand ils ont parfois recours à des *« middlewares »*, il s'agit d'outils plus limités en terme de puissance technique, car moins onéreux. De plus, du fait d'équipes réduites, le Game Designer professionnel indépendant est impliqué dans de plus nombreuses phases de la création du jeu *(programmation, graphismes, son, création de niveaux…)* que son homologue de l'industrie. Il se doit donc, soit d'utiliser des outils moins complexes, soit d'améliorer son niveau de formation pour participer aux différentes étapes de la création vidéoludique.

Malgré leur indépendance et le fait qu'ils utilisent parfois des outils différents, les Game Designers indépendants restent cependant assez proches des professionnels de l'industrie. Il arrive même que des professionnels de l'industrie choisissent de devenir indépendants. Exemple original de Serious Game sur ce thème, *A Message for 2K Australia* ***(Jarrad Woods, 2009)*** a été créé par l'employé d'un studio de développement de jeu vidéo pour annoncer sa démission. Le jeu en lui-même reprend des séquences du classique *Super Mario Bros.* ***(Nintendo, 1985)*** agrémenté de textes de l'auteur. Il explique qu'après avoir passé de nombreuses années dans l'industrie, il a maintenant envie de retrouver une liberté créative, et se tourne donc vers la sphère indépendante. Dès que le joueur arrive à terminer un des tableaux du jeu, le message *« I Quit ! » (« Je démissionne ! »)* clignote sur l'écran avant d'afficher le tableau suivant. Ce Game Designer a explicitement créé un Serious Game pour annoncer et justifier sa démission, tout en faisant de la publicité pour lancer sa nouvelle carrière. Cette même stratégie a été employée quelques mois plus tard par ***William David***, un employé d'***UbiSoft***. Il a réalisé le jeu *Leaving* ***(William David, 2009)*** pour annoncer sa démission de l'entreprise afin d'embrasser une carrière d'indépendant. Cette proximité entre le Game Designers de l'industrie et les indépendants fait que les approches de facilitation du Game Design dont ils bénéficient sont très proches.

Notons également que, d'un point de vue historique, ces créateurs de jeux indépendants ont joué un rôle important dans les approches de facilitation du Game Design. D'une manière analogue à l'industrie du cinéma reposant sur une symbiose entre majors et indépendants (Blanchet, 2010), les créateurs professionnels indépendants du jeu vidéo expérimentent des pratiques qui sont ensuite récupérées par l'industrie. Un exemple de ces pratiques est le fait de mettre à la disposition des joueurs un ensemble d'outils leur permettant de modifier les jeux vidéo réalisés par des professionnels. Expérimentée et encouragée par les pionniers du *shareware* (***id Software***, ***Apogee***, ***Epic Megagames***…) dans les années 1990, cette idée a aujourd'hui donné naissance au *« modding »*, sur lequel nous reviendrons ci-après *(p.50)*. Autre exemple, l'apparition du *« middleware »* est également imputable à des indépendants. Tel qu'observé par ***De Prato & al.*** (2010), la société ***id Software*** a initié ce mouvement avec la commercialisation des technologies de moteur de jeu utilisées pour créer *Doom* ***(id Software, 1993)*** et *Quake* ***(id Software, 1996)***. En vendant à des sociétés concurrentes une technologie alors à la pointe, cette petite société indépendante a inventé une pratique aujourd'hui courante dans l'industrie vidéoludique.

3.2 LES CREATEURS AMATEURS

Contrairement aux professionnels, les créateurs amateurs ne distinguent pas à travers leur cadre de travail, mais par les outils qu'ils utilisent. D'un côté, certains amateurs s'appuient sur les outils livrés avec des jeux commerciaux, réalisés par des professionnels, pour créer des variantes de ces jeux. Qu'il s'agisse de la création de nouveaux niveaux, de la modification de règles ou du remplacement de graphismes, les créations associées à ces *« outils de modding »* ne sont pas autonomes : il faut posséder le jeu originel pour les utiliser. D'autres amateurs se tournent donc vers un autre type d'outil, qui permet de créer de nouveaux jeux vidéo autonomes : les *« usines à jeux »*.

Notons également que, si les professionnels ont à leur disposition des approches de facilitation d'ordre théorique *(méthodologie…)* et technique *(logiciels…)*, les amateurs ont uniquement recours à des approches de facilitation d'ordre technique.

3.2.1 LES USINES A JEUX

La pratique amateur du Game Design est intimement liée à l'histoire de l'informatique, en particulier à celle du développement de la micro-informatique personnelle. Une des premiers micro-ordinateurs à avoir rencontré un succès commercial est l'*Altair 8800* ***(MITS, 1975)***. Cet ordinateur avait la particularité d'être vendu en kit. Il a donc rapproché de nombreux passionnés d'informatique, qui se sont regroupés au sein de « clubs » afin de s'entraider pour les assembler. Ces clubs constituaient également un moyen d'échanger des programmes pour cette machine, en particulier de nombreux petits jeux vidéo écrit en *BASIC* (Wolf, 2007). Réalisés par des amateurs, ces jeux vidéo s'échangeaient directement sous forme de listing *BASIC*. Cette pratique informelle a rapidement été accompagnée par la publication d'ouvrages regroupant des codes sources de jeux en *BASIC*, à l'image de *Basic Computer Games* (Ahl, 1978). Certains magazines se spécialisèrent même dans cette activité, comme le français *Hebdogiciel* ***(Shift Éditions, 1983-1987)***, qui publiait de nombreux codes sources de jeux. L'arrivée de l'*Apple II* ***(Apple, 1977)***, équipé d'un lecteur de disquettes, permis enfin aux amateurs d'échanger leurs créations vidéoludiques directement sous forme électronique.

Cependant, devant la complexité que représente la création d'un nouveau jeu vidéo sous forme électronique, certains programmeurs ont imaginé des outils logiciels permettant de faciliter leur création. Le plus ancien que nous ayons pu recenser à ce jour est *Eamon* ***(Donald Brown, 1980)***. Explicitement destiné à une diffusion non commerciale *(ancêtre du « freeware »)*, ce programme est un jeu de rôle et d'aventure en mode texte, dans la grande tradition de *Colossal Cave Adventure* ***(William Crowther & Don Woods, 1976)***. Mais surtout, il est livré avec des utilitaires permettant de créer ses propres aventures, qui sont ensuite redistribuables sous forme de jeux autonomes (Kunze & Giola, 2008). Il s'agit là d'un exemple de programme destiné à simplifier la création de nouveau jeux vidéo, de manière à la rendre accessible à des non-programmeurs. Ce genre d'outil logiciel s'est considérablement développé par la suite, structurant la sphère des Game Designers amateurs en « communautés » centrées sur un logiciel donné. **Nous appelons de tels logiciels des « usines à jeux », que nous pouvons définir comme un logiciel « tout-en-un » permettant de créer un jeu vidéo autonome sans forcément partir d'une base existante**.

Ces outils se rapprochent donc de certains *« middleware »* utilisé par les professionnels, en particulier des *« middleware »* reposant sur une solution « tout-en-un ». Qu'est-ce qui différencie alors une *« usine à jeux »* d'un *« middleware »* ? Tout simplement son niveau d'accessibilité. Là où l'utilisation d'un *« middleware »* nécessitera de posséder un savoir technique conséquent, la manipulation d'une usine à jeux devra être suffisamment simple pour être accessible à des créateurs amateurs, qui ne possèdent généralement aucune formation particulière liée à la création vidéoludique *(p.14)*. Malgré l'intérêt que représente ce type de logiciel, la littérature scientifique qui s'y rapporte est quasi-inexistante. Une des rares chercheuses à aborder le sujet est ***Sihvonen*** (2009). Dans sa thèse portant sur le *« modding »* lié à la série des *Sims*, elle évoque le lien de parenté évident entre les « outils de modding » et les « usines à jeux » des années 1980. Conséquence de cette littérature très réduite, il n'existe pas à notre connaissance de listing exhaustif recensant les usines à jeux. Pourtant, ces outils techniques représentent une approche de facilitation de la création vidéoludique qui nous semble particulièrement pertinente pour répondre à notre problématique. En effet, contrairement aux approches de facilitation du Game Design destinées aux professionnels, les « usines à jeux » semblent accessibles à une large variété de public. Nous proposons donc

d'esquisser un premier travail de recensement des usines à jeux, avant d'étudier leur potentiel pour faciliter la création de Serious Games *(p.166)*.

3.2.2 LES OUTILS DE « MODDING »

Si les amateurs qui utilisent une « usine à jeux » cherchent à créer de nouveaux jeux autonomes, de nombreux autres créateurs préfèrent modifier ou créer du contenu pour un jeu existant. Certains créateurs professionnels proposent des outils permettant à tout joueur de « modifier » le jeu qu'ils ont créé. Nous désignerons ces outils par l'appellation générale *« outil de modding »*, la notion de *« modding »* étant usuellement employée pour designer le fait de modifier un jeu existant. **Les « outils de modding » sont donc tout simplement un ensemble d'éditeurs logiciels permettant de modifier les différents aspects d'un jeu vidéo**. Ces outils sont utilisés de deux manières différentes. D'un côté, de nombreux amateurs utilisent uniquement des *« éditeurs de niveaux »* pour créer ou modifier les niveaux d'un jeu. De l'autre, d'autres créateurs ont recours à plusieurs outils pour créer un *« mod »*, autrement dit une variante d'un jeu donné.

3.2.2.1 Les éditeurs de niveaux

L'outil le plus populaire pour modifier des jeux existants est sans conteste **« l'éditeur de niveaux »**. **Un « éditeur de niveau » est un outil qui permet d'inventer de nouvelles situations de jeu pour un titre existant.** Concrètement, un « éditeur de niveau » permet au concepteur d'agencer des « éléments de jeu », tels que des ennemis ou des bonus, dans un espace afin de créer une situation de jeu. La notion de « niveau » varie bien entendu considérablement selon le type de jeu : *dans un jeu de course il s'agira d'un circuit, dans un jeu d'aventure d'un scénario, dans un jeu de plateforme d'un ensemble d'obstacles...*

Le premier exemple d'éditeur de niveaux présent dans un jeu vidéo commercial semble être celui livré avec *K.C. Munchkin!* ***(Ed Averett, 1981)***. Ce titre est un clone de *Pac-Man* ***(Namco, 1980)***. Son éditeur de niveaux permet aux joueurs de créer leurs propres labyrinthes sous forme graphique : il leur suffit de disposer des blocs à l'écran pour créer un nouveau niveau de jeu. Au delà de cet exemple pionnier, le premier jeu vidéo a avoir été plébiscité pour son éditeur de niveaux est *Lode Runner* ***(Douglas Smith, 1983)***. A l'origine, le concepteur de ce titre avait créé un outil pour l'aider à réaliser les 150 niveaux de son jeu. Cet éditeur de niveaux affiche une liste de différents « blocs » pouvant être librement agencés à l'écran pour construire un ensemble de plateformes, d'obstacles, d'ennemis et d'objets à récolter. Cet outil n'était pas destiné à être diffusé, mais devait uniquement servir à des fins de création interne. Cependant, lors de la réalisation du jeu, ***Douglas Smith*** se trouva rapidement à cours d'inspiration pour créer des niveaux. Afin de livrer les 150 niveaux en temps et en heure à son éditeur, il sollicita l'aide des enfants du voisinage. Il leur demanda ainsi d'inventer des niveaux en utilisant son éditeur. Remarquant alors la façon dont les enfants s'amusaient à créer des niveaux avec cet outil simple d'utilisation, il décida de l'inclure en cadeau avec le jeu. De cette décision motivée par un planning serré, est né un jeu qui conservera une place dans l'histoire pour avoir permis à ses joueurs de le modifier (Gillet & Gorges, 2008). Cet exemple illustre parfaitement le niveau de facilitation de la création vidéoludique proposé par cette approche. Si des enfants ont pu utiliser sans problème cet « éditeur de niveau », c'est qu'il s'agit d'une démarche pertinente pour mettre la création vidéoludique à la portée de novices.

Pourtant, nous avons tout à l'heure mentionné que la création de niveaux est également un métier de l'industrie, désignée par l'expression de *« Level Design »*. Comment un même outil, l'éditeur de niveaux, peut-il être à la fois accessible à des enfants et représenter un métier industriel nécessitent une formation professionnelle ? Tout simplement parce qu'il existe des éditeurs de niveaux plus ou moins puissants, et donc plus ou moins complexes à manipuler. De nos jours, de nombreux jeux sont livrés avec des éditeurs de niveaux. Ces derniers sont

parfois des versions allégées des outils utilisés pour créer le jeu, à l'image de *UnrealED* pour la série des jeux *Unreal* ***(Epic Games, 1998-2007)*** ou de *The Elder Scroll Construction Set* pour les jeux *Morrowind* ***(Bethesda Softworks, 2002)*** et *Oblivion* ***(Bethesda Softworks, 2006)***. Ces outils étant originellement conçus pour des professionnels, leur manipulation nécessite souvent un petit apprentissage, surtout depuis l'avènement de la 3D. Une autre piste consiste donc à créer des éditeurs de niveaux plus simples *(mais aussi plus limités)*, qui pourront être appréhendés plus facilement par les amateurs. La dernière génération de consoles accueille des titres *« AAA »* proposant de tels éditeurs de niveaux (Sotamaa, 2010). Par exemple, *LittleBigPlanet* ***(MediaMolecule, 2008)*** est livré avec un éditeur de niveau, le jeu *Halo 3* ***(Bungie, 2007)*** dispose d'un outil baptisé *Forge*, alors que *Smash Brosh Melee* ***(Nintendo, 2008)*** propose son *Stage Builder*. Cependant, tous les logiciels permettant de créer de nouveaux niveaux ne sont pas forcément le fruit des créateurs originels du jeu. Il arrive que certains amateurs créent eux-mêmes des outils officieux permettant d'éditer des niveaux. Par exemple, la série des *Tomb Raider* ***(Core Design, 1996-2000)***, a vu apparaître un outil du nom de *Dxtre3D* ***(Turbo Pascal, 1997)***. Ce logiciel fut le seul éditeur de niveau disponible pendant trois ans, jusqu'à ce que ***Core Design*** décide de créer un outil officiel, *Tomb Raider Level Editor* ***(Core Design, 2000)***, pour la sortie du cinquième volet de la série.

D'un point de vue quantitatif, il n'existe malheureusement pas à ce jour de recensement exhaustif des éditeurs de niveaux. A titre d'indication, le site *MobyGames*, qui ne référence que les outils officiels livrés avec des jeux commerciaux, référence 1246 éditeurs de niveaux[21]. Il faut ajouter à ce chiffre les éditeurs de niveaux officieux, les éditeurs de niveaux pour les jeux non-commerciaux, et surtout les nombreux éditeurs de niveaux professionnels qui ne sont pas distribués au grand public. Nous pouvons alors supposer qu'il existe un nombre relativement conséquent d'éditeurs de niveaux pour les jeux vidéo de divertissement.

3.2.2.2 Les SDK pour la création de « mods »

Aussi pertinent et populaire que soient les éditeurs de niveaux, leur portée reste néanmoins limitée. En effet, ce type d'outil ne permet généralement pas de modifier les règles d'un jeu, ni de transformer son apparence graphique ou sonore. Certains créateurs amateurs ont donc recours à d'autres « éditeurs » dédiés aux différents aspects du jeu. On nomme l'ensemble des outils permettant de modifier un jeu commercial un **« Software Development Toolkit »** *(SDK)*. Par exemple, le *Source SDK* permet de modifier tout jeu basé sur la technologie *Source Engine*, de *Half-Life 2* ***(Valve Software, 2004)*** à *Left 4 Dead 2* ***(Valve Software, 2009)***. Ce SDK est composé d'une vingtaine d'outils tels que: *Hammer Editor*, un éditeur de niveaux ; *Face Poser* et *Vtex*, qui permettent de modifier l'apparence des éléments du jeu ; ainsi qu'une série de scripts permettant de modifier les règles du jeu. En général, de tels SDK, distribués par le créateur originel du jeu, sont basés sur les outils utilisés pour développer le titre, bien qu'ils soient parfois volontairement bridés pour simplifier leur utilisation.

En utilisant un SDK, les créateurs amateurs peuvent modifier l'intégralité d'un jeu vidéo : *graphismes, sons, règles…* Par exemple, *Aliens TC* ***(Justin Fisher, 1994)*** transforme le jeu *Doom* ***(id Software, 1993)*** en un jeu d'action basé sur le film *Aliens* ***(James Cameron, 1986)***, tandis que *Batman Doom* ***(ACE Team Software, 1997)*** permet d'incarner le célèbre super-héros dans un titre qui ne ressemble plus vraiment au jeu originel. Si le simple fait de créer de nouveaux « niveaux » pour un jeu donné ne possède pas d'appellation particulière, une telle modification des autres aspects d'un jeu vidéo correspond à ce que les joueurs appellent un **« mod »**. Comme son nom l'indique**, un « mod » est un ensemble de modifications pour un jeu donné, présenté sous forme redistribuable**. Chacun est ainsi libre de récupérer un *« mod »* et, pour peu qu'il possède le jeu original, de l'utiliser afin de jouer au jeu modifié (Bogacs, 2008). Cette pratique est aujourd'hui fort répandue, popularisé par des créations

[21] Relevé le 09-01-2011 à partir de http://www.mobygames.com/genre/sheet/editor-construction-set/

telles que *Counter-Strike* ***(Minh Le & Jess Clife, 1999)***, qui est un « mod » du jeu *Half-Life* ***(Valve Software, 1998)*** ainsi que *Defense Of The Ancients* ***(Eul, 2003)***, un « mod » pour *Warcraft III* ***(Blizzard Entertainment, 2002)***. Nous identifions même des Serious Games réalisés sous forme de « mod », à l'image de *Escape from Woomera* *(p.25)*.

Comme pour les autres outils utilisés par les Game Designers amateurs, la création de « mods » s'articule souvent au travers de communautés (T. Taylor, 2009). Un SDK étant fourni avec un titre donné, l'éditeur du jeu essaie alors de prévoir un espace dédié aux échanges de « mods » sur le site officiel du jeu. C'est par exemple le cas de *Dawn of War II* ***(Relic Entertainment, 2009)***[22]. Cependant, en l'absence de SDK officiel, les amateurs sont prêts à « hacker » le jeu pour créer leurs propres outils. Ainsi, les nombreux outils de modding disponibles pour la série des *Sims* ***(Maxis, 2000-2009)*** co-existent avec des outils officiels très limités et une plateforme d'échange officielle (Sihvonen, 2009). Les plateformes d'échanges peuvent également être officieuses, à l'image du site *Mod DB* ***(2002-2010)*** qui héberge 7.518 mods pour un large panel de jeux vidéo[23]. Du point de vue des pratiques communautaires, les « mods » et la création de niveaux sont donc similaires (Knorr, 2007). En effet, créer un niveau consiste déjà à créer des « modifications redistribuables » pour un jeu donné. Mais un simple niveau ne serra pas qualifié de « mod » par les Game Designers amateurs. Cela est sûrement du au fait qu'un niveau ne modifie pas suffisamment le jeu vidéo originel pour le considérer comme « différent ». En effet, dans les communautés de *« moddeurs »*, il est d'usage de distinguer les *« Partial Conversion »* et les *« Total Conversion »*. Cette différenciation est basée sur la quantité d'éléments du jeu originel qui est modifiée (Sotamaa, 2009). On parle de *« Total Conversion »* lorsqu'il n'est plus possible de reconnaître le jeu originel. Historiquement, le mod *Aliens TC* pour *Doom* a vraisemblablement été le premier à revendiquer cette appellation (Bogacs, 2008).

Si une telle catégorisation s'opère sur la quantité de travail investie dans les modifications, c'est parce que la création de « mods » est loin d'être une tâche simple. Si les éditeurs de niveaux restent des outils relativement accessibles, les autres éditeurs d'un SDK sont loin d'être aussi aisés à prendre en main. Plus particulièrement, la création de nouveaux graphismes requiert d'utiliser des logiciels professionnels de graphisme *(modélisation 3D, textures…)*. De même, la modification des règles de jeux nécessite d'utiliser un langage de script spécifique au jeu, voire un langage commun tel que *C++*. La création de « mods » de qualité requiert donc une connaissance technique qui s'approche de celle des professionnels, assortie d'un investissement en temps conséquent. Il est ainsi courant de voir les « moddeurs » travailler en équipe spécialisée *(graphistes, programmeurs, level designers…)*. La contrepartie de cette accessibilité relativement restreinte est que la création de « mod » constitue une excellente passerelle vers l'industrie du jeu vidéo pour des amateurs. Par exemple, ***Tim Willits*** a pu intégrer ***id Software*** en tant que Game Designer grâce au succès rencontrés par les mods et niveaux pour *Doom* qu'il créa durant son temps libre (Kushner, 2004). Dans un registre similaire, le célèbre level designer ***Richard « Levelord » Gray*** fut embauché par ***3D Realms*** grâce aux niveaux pour *Doom* qu'il réalisa en tant que hobby, alors qu'il exerçait déjà le métier de programmeur dans l'industrie aéronautique (Djaouti, 2010b, 2010c). Une étude illustre même que la création de « mods » représente une valeur marchande indirecte pour les éditeurs de jeux vidéo (Postigo, 2007). En effet, de part leur nature gratuite conditionnée par la nécessité de posséder le jeu original pour y jouer, les « mods » permettent de prolonger la vie commerciale du titre sur lequel ils se basent. Que ce soit pour promouvoir leurs réalisations ou pour recruter de nouveaux talents, certains studios professionnels de création vidéoludique incorporent donc la création amateur de « mod » dans leur stratégie commerciale (Nieborg & van der Graaf, 2008).

[22] Retrouvé le 01-12-09 sur http://community.dawnofwar2.com/forums/world-builder-and-modding
[23] Retrouvé le 07-02-11 de http://www.moddb.com/mods

Enfin, une dernière approche existe pour permettre à tous les joueurs de participer à la création vidéoludique. Nombre de jeux vidéo proposent, par le biais d'un « menu d'options », de configurer des paramètres influant sur le déroulement de la partie. La plupart du temps, ces menus proposent aux joueurs de configurer les touches et boutons utilisés pour jouer, ou à défaut permettent de choisir entre plusieurs « modes de contrôles ». Par exemple, dans *Mario Kart Wii* ***(Nintendo, 2008)***, chaque joueur peut choisir entre trois périphériques de contrôle : *« Wiimote+Nunchuk »*, *« Wiimote+volant »* ou *« manette de GameGube »*. Les joueurs peuvent également choisir entre les styles de conduite *« automatique »* ou *« manuel »*. En style *« manuel »*, les joueurs ont la possibilité de déclencher des accélérations « turbo » en réalisant des dérapages. Cette « règle de jeu » est remplacée par une plus grande capacité de braquage avec le style *« automatique »*. Une observation similaire peut être faite avec les jeux pour ordinateur. Par exemple, *Lego Indiana Jones* ***(Traveller's Tale, 2008)*** permet au joueur de choisir quelles sont les touches du clavier qu'il utilisera pour contrôler son personnage. De plus, certains titres comme *Left 4 Dead* ***(Turtle Rock Studios, 2008)*** permettent aux joueurs de choisir un « niveau de difficulté ». Ce réglage affecte à la fois les règles du jeu et l'organisation des niveaux : un réglage de difficulté élevé se traduit par l'augmentation du nombre d'ennemis et de leur puissance. Enfin, certains titres permettent de configurer l'apparence graphique et sonore du jeu. Par exemple, dans *Quake 3* ***(id Software, 1999)***, les joueurs peuvent modifier l'angle de vision, les informations affichées à l'écran ou encore l'apparence de leurs avatars. Cette fonctionnalité est particulièrement appréciée des joueurs professionnels. Ces derniers configurent l'affichage du jeu durant les compétitions de manière à ne garder à l'écran que les informations *« utiles à la victoire »*, telles que l'on montré des études sur le sujet (Connor, Fiske, & Kennedy, 2006; Rouchier & Retaux, 2002).

Le principal avantage de ces menus d'options est bien évidement leur simplicité d'utilisation : toute personne capable de choisir un élément dans une liste est potentiellement capable de les utiliser. Cependant cette accessibilité se fait au prix de la liberté de création : les joueurs ne pourront rien modifier d'autre que ce qui aura été prévu par les concepteurs du jeu. Pour tenter de pallier cet aspect, certains concepteurs de jeux proposent un très grand nombre de paramètres à configurer. Par exemple, les menus d'options de *Worms 2* ***(Team 17, 1997)*** permettent de modifier littéralement toutes les valeurs numériques des règles du jeu, du diamètre de l'onde de choc produit par chaque arme à la fréquence d'apparition des bonus, en passant par le niveau de vie des avatars. L'augmentation du nombre de choix proposés par ces outils ne les transforme pas pour autant en *« éditeurs de contenu émergeant »* *(p.160)*. En les utilisant, un Game Designer amateur ne pourra rien créer de plus que ce qui aura déjà été inventé par le créateur originel du jeu. C'est pour cela que, culturellement, les « menus d'options » ne sont pas véritablement considérés comme des outils de création vidéoludique, mais simplement comme une possibilité de personnalisation de l'expérience de jeu. Il ne faut pourtant pas oublier qu'ils permettent bel et bien de modifier tous les éléments d'un jeu vidéo, des règles aux modes de commandes, malgré le cadre très limité qu'ils proposent. A défaut d'être une catégorie d'outil de création vidéoludique à part entière, les « menus d'options » représentent donc un premier pas vers la modification de jeux vidéo.

4 SYNTHESE : PLUSIEURS APPROCHES DE FACILITATION DU GAME DESIGN

Dans ce chapitre, nous avons abordé de nombreux concepts liés à la création de jeu. Si le *« Game Design »* renvoie normalement à des jeux sur tout type de support *(p.44)*, dans cette thèse **nous nous focaliserons uniquement sur les jeux vidéo.** En effet, nous avons déjà fait le choix d'étudier uniquement des Serious Games sur support informatique, qui caractérisent la « vague actuelle » *(p.18)*. En ce qui concerne la notion de « jeu », nous nous appuyons sur le *modèle classique du jeu* proposé par ***Juul***, cette définition synthétisant plusieurs définitions antérieures *(p.42)*.

Nous avons alors rencontré **trois définitions possibles du terme « Game Design »** :

- Le processus global de création d'un jeu *(p.42)*.
- La phase de conception d'un jeu, distincte de celle de fabrication *(p.44)*.
- Une partie de la phase de conception : la création des mécanismes et de l'univers de jeu. Cette partie se distingue de celle du *« Level Design »*, qui correspond à la construction des niveaux du jeu *(p.45)*.

Nous choisissons de nous inscrire dans la définition du *« Game Design »* comme processus global de création d'un jeu vidéo. Sauf mention contraire, dans cette thèse la notion de « Game Design » renvoie donc aussi bien à la conception qu'à la fabrication d'un jeu vidéo, et englobe également la notion de *« Level Design »*.

Nous nous sommes ensuite intéressé aux approches de facilitation du *« Game Design »*. Nous constatons que le *« Game Design »* peut être pratiqué par plusieurs types de personnes : des créateurs professionnels, des amateurs, et même par n'importe quel joueur de jeu vidéo. En étudiant les approches de facilitation de la création vidéoludique destinées à ces trois profils, nous recensons d'un côté des **outils théoriques**, *liés à la phase de conception*, et des **outils techniques**, *liés à la phase de fabrication (p.44)*.

Du côté des outils théoriques, nous avons uniquement recensé des approches de facilitation visant les professionnels. Ces derniers ont à leur disposition des *méthodologies de conception* et autres *outils d'aide à la réflexion créative (p.46)*. Du côté des outils techniques, nous recensons tout d'abord le **« middleware »** que nous définissons comme *un outil logiciel permettant d'accélérer la création de jeux vidéo*. Destinés aux professionnels, les *« middlewares »* impliquent généralement de posséder un certain niveau de compétences techniques pour être utilisés *(p.46)*. A l'inverse, nous identifions les **« menu d'options »**, que nous définissons comme *un outil logiciel permettant de configurer des paramètres influant sur le déroulement de la partie*. Suffisamment simple pour être utilisée par n'importe quel joueur, cette approche limite grandement le potentiel créatif à des fins d'accessibilité *(p.53)*. Nous identifions alors des outils techniques destinés aux amateurs qui semblent offrir un bon compromis entre accessibilité et possibilités de création. Tout d'abord, les **« usines à jeux »** que nous définissons comme *un outil logiciel permettant de créer un jeu vidéo autonome sans partir d'une base existante (p.49)*. Nous recensons également les **« outils de modding »** que nous définissons comme *un ensemble d'outils logiciels permettant de modifier les différents aspects d'un jeu vidéo (p.50)*. Qu'ils permettent de créer de nouveaux jeux ou de modifier des titres existants, ces outils techniques facilitant la création vidéoludique sont basés sur le même principe. Ils proposent une collection *« d'éditeurs »* permettant de créer ou modifier un aspect précis d'un jeu vidéo. Par exemple, les **« éditeurs de niveaux »** permettent *d'inventer des situations de jeu* de manière visuelle, ce qui les rend particulièrement accessibles *(p.50)*.

BILAN DE LA PARTIE I : CADRE D'ANALYSE DU SERIOUS GAME ET DU GAME DESIGN

1 UNE APPROCHE DU SERIOUS GAME POUR L'ETUDE DE SA CONCEPTION

Comme nous l'avons exploré dans ce chapitre, la notion de « Serious Game » renvoie à plusieurs approches différentes. Si elle se caractérise systématiquement par le fait d'associer une dimension « ludique » et une dimension « sérieuse », nous avons identifié de nombreux supports, thématiques, finalités et époques qui la caractérisent *(p.18)*. L'étude approfondie du « Serious Game » représente donc une thématique de recherche à part entière (J. Alvarez & Djaouti, 2010), qui implique l'analyse de son histoire, la rencontre des différents types d'acteurs présents dans le champ ou encore une exploration détaillée des différents sujets qu'il aborde *(p.32)*. Aussi intéressante soit-elle, l'étude globale du phénomène des « Serious Games » n'est pas l'objet de cette thèse. Cette étude préliminaire vise cependant à définir notre contexte de recherche. Face aux différentes approches possibles de cet objet, nous choisissons de nous inscrire au sein de la « vague actuelle » des Serious Games *(p.18)*. Nous nous intéressons donc à **l'étude des Serious Games sur support vidéoludique** pour toutes les finalités et marchés *« autres que le simple divertissement »* identifiés par le *modèle G/P/S (p.28)*.

Dans ce contexte, nous identifions plusieurs travaux et projets relatifs à la conception de Serious Games *(p.39)*, dont les objectifs semblent se rapprocher de notre problématique *(p.14)*. Ces quelques « tendances actuelles » du secteur partagent de nombreux points communs. Tout d'abord, elles visent toutes à simplifier la création de Serious Games. Mais surtout, que ce soit pour réduire les coûts au sein de l'industrie ou pour démocratiser la création de jeux par des enseignants, elles reposent toutes sur des outils logiciels permettant de modifier ou de créer des Serious Games. Le concepteur de jeu ***Richard Rouse*** met d'ailleurs en évidence l'importance des outils dans la création industrielle de jeux vidéo de divertissement (Rouse, 2000). Pour autant, quelle que soit la facilité de prise en main ou la puissance d'un outil donné, son utilisation implique généralement un minimum de connaissances en conception vidéoludique pour aboutir à une réalisation concrète. Ainsi, les propos de ***Richard Rouse*** sur le rôle des outils ne constituent qu'un des vingt-six chapitres de son ouvrage *Game Design - Theory & Practice* (Rouse, 2001). Comme le laisse supposer le titre de cet ouvrage, les autres chapitres sont destinés à transmettre des connaissances théoriques ou appliquées sur le processus de conception d'un jeu vidéo. Ce savoir professionnel relatif à la création de jeu, qu'il s'agisse de la conception théorique ou de la manipulation pratique d'outils est désigné par le terme de *« Game Design »*. Pour explorer la question de la création de Serious Game, il nous semble donc pertinent de commencer par nous intéresser au champ du *« Game Design »*. Bien que ce dernier soit normalement associé au secteur du divertissement, les tendances actuelles du secteur du Serious Game nous permettent d'imaginer y identifier d'éventuelles réponses à notre problématique.

2 APPLICATION DU GAME DESIGN AUX SERIOUS GAMES : LE SERIOUS GAME DESIGN

Nous avons défini le *« Game Design »* comme le processus de création d'un jeu *(p.54)*. D'une manière analogue, nous pourrions **définir le « Serious Game Design », comme le processus permettant de créer un Serious Game**. Nous pouvons alors nous interroger sur la pertinence d'appliquer des approches du *« Game Design »* au *« Serious Game Design »*. Dans sa thèse, ***Alvarez*** (2007) analyse le phénomène des Serious Games selon trois approches : *culturelle*, *pragmatique* et *formelle*. Ces trois « niveaux de lecture » sont inspirés du modèle théorique de ***Salen & Zimmerman*** *(p.75)*. En conclusion de son travail pionnier visant à caractériser le Serious Game, ***Alvarez*** note :

> *« Ainsi, nous pensons qu'il n'y aurait pas […], sur le plan informatique, de caractéristiques propres au Serious Game. Cette catégorie n'existerait donc pas d'un point de vue formel, et serait, de ce fait, à considérer uniquement au niveau d'un système culturel ou pragmatique, c'est-à-dire comme un genre de jeux vidéo. » [p.265]*

Ainsi, d'un point de vue conception, le Serious Game ne se distinguerait pas du jeu vidéo sur le plan technique, mais plutôt sur la dimension théorique. A la lumière de ces réflexions, nous pourrions alors émettre l'hypothèse que les approches du *« Game Design »* seraient bien applicables au *« Serious Game Design »*, au moins au niveau formel. Nous proposons alors d'**étudier les approches de formalisation du « Game Design », puis d'évaluer leur pertinence pour la conception de Serious Games**.

Mais les conclusions d'***Alvarez*** laissent également supposer que le *« Serious Game Design »* se différencie du *« Game Design »* par certains aspects liés aux niveaux culturel et pragmatique. Par exemple, un Serious Game est explicitement conçu pour servir une finalité « sérieuse ». Son concepteur doit donc non seulement veiller à créer un jeu « amusant », mais également s'assurer que le jeu serve effectivement une vocation utilitaire donnée. Nous pouvons alors supposer que le processus de création d'un Serious Game n'est pas identique à celui de création d'un jeu vidéo de divertissement. Nous proposons donc également d'**étudier globalement le processus de conception de Serious Games afin de caractériser ses spécificités par rapport à la conception de jeux vidéo de divertissement**.

Tel que nous l'avons déjà évoqué *(p.14)*, ces deux thématiques de recherche sont induites par notre problématique. Pour identifier les approches permettant de faciliter la création de Serious Games, il convient d'étudier conjointement le potentiel des approches issues du « Game Design » pour faciliter leur création tout en analysant les différences entre le processus de création d'un jeu vidéo de divertissement et d'un Serious Game.

3 DISTINCTION ENTRE OUTILS THEORIQUES ET OUTILS TECHNIQUES

Notre travail de recherche repose sur l'étude croisée du *« Game Design »* et du *« Serious Game Design »*. Lors de notre étude générale du *« Game Design »*, nous avons pu observer différentes approches de facilitation qui lui sont associées *(p.46)*. Nous observons tout d'abord des approches de facilitation s'appuyant sur des **outils « théoriques »**, en la forme de méthodologies visant à guider la réflexion créative du concepteur. Nous identifions ensuite des approches mobilisant des **outils « techniques »**, qui permettent de fabriquer les jeux vidéo sous forme de programme informatique. Bien que toutes les approches de facilitation du *« Game Design »* soient potentiellement intéressantes pour le *« Serious Game Design »*, nous ne pourrons pas toutes les explorer dans cette thèse. Nous proposons donc de sélectionner les outils présentant les approches de facilitation les plus pertinentes par rapport à notre problématique. Nous avons déjà évoqué les différences de niveau d'accessibilité et de possibilité créative offerte par les approches identifiées *(p.54)*. Pour répondre à notre problématique *(p.14)*, nous nous focaliserons sur les outils de chaque domaine *(théorique et technique)* qui semblent offrir le meilleur compromis entre facilité d'utilisation et possibilité de création :

- **Les « outils théoriques » utilisés par les professionnels du jeu vidéo**. Les cultures ancrées dans les pratiques amateurs du Game Design ne possèdent pas véritablement de savoir théorique formalisé, et encore moins de formation ou littérature spécifique. Nous étudierons donc uniquement les « outils théoriques » destinés aux professionnels de l'industrie, et analyserons dans quelle mesure ils sont applicables au secteur du Serious Game.

- **Les « outils techniques » utilisés par les Game Designers amateurs**, et plus particulièrement les *« usines à jeux »*. Les outils utilisés par les professionnels requièrent généralement un niveau de compétence technique élevé *(p.46)*, les rendant moins pertinents que ceux des amateurs pour notre problématique. Au sein des pratiques amateur, celle du *« modding »* attire particulièrement notre attention, d'autant plus que nous recensons déjà des exemples de « Serious Games » créés sous forme de *« mods »* *(p.25)*. Cependant, l'utilisation de *« SDK »* pour modifier des jeux vidéo existants nécessite un niveau de compétence relativement élevé, en particulier si les concepteurs souhaitent créer de nouveaux graphismes ou de nouvelles règles de jeu *(p.51)*. Il existe certes des outils très simples d'accès comme les *« éditeurs de niveaux »*, mais ces derniers ne permettent pas de travailler sur tous les aspects d'un jeu vidéo *(p.50)*. Au final, les *« usines à jeux »* nous semblent être les outils techniques les plus pertinents par rapport à notre problématique. Etant pensés pour des amateurs, ils sont généralement simples d'utilisation. Et contrairement aux outils de *« modding »*, ils permettent de créer de nouveaux jeux autonomes, sans être limités à la modification de jeux vidéo existants *(p.49)*.

Nous étudierons de manière séparée ces deux groupes d'outils, afin d'identifier des approches de facilitation pouvant être pertinentes pour le Serious Game. Cette distinction nous pousse également à étudier de manière séparée les aspects « théoriques » et « techniques » du *« Serious Game Design »*. Que ce soit pour l'analyse des outils du *« Game Design »* ou l'étude plus générale du *« Serious Game Design »*, nous **distinguerons donc les considérations d'ordre théorique des considérations d'ordre technique**. La deuxième partie de cette thèse est consacrée à l'étude des considérations théoriques *(p.58)*, tandis que la troisième partie s'attache à l'analyse des considérations techniques *(p.155)*.

[PARTIE II]
CONSIDERATIONS THEORIQUES SUR LE SERIOUS GAME DESIGN

Cette partie est consacrée à l'analyse de l'aspect théorique de la création de Serious Games. Nous nous intéresserons tout d'abord aux outils théoriques issus du « Game Design » *(p.59)*. Nous en identifions deux types : les méthodologies de conception *(p.60)*, et les outils d'aide à la réflexion créative *(p.68)*. Après avoir étudié ces outils théoriques destinés aux concepteurs professionnels de jeux vidéo de divertissement, nous chercherons à identifier ceux qui sont spécifiquement destinés à faciliter la création de Serious Games. Ces derniers sont principalement des méthodologies de conception *(p.86)*. Suite à leur analyse, nous serons en mesure de proposer un modèle générique du processus de création d'un Serious Game : le *modèle générique DICE (p.103)*.

La comparaison des outils théoriques issus du « Game Design » avec ceux du « Serious Game Design » fait ressortir quelques différences liées à la conception de ces deux catégories de jeux vidéo. Nous étudierons alors plus en détail les spécificités théoriques de la création de Serious Games en nous appuyant sur des retours d'expérience du *CIFRE (p.106)*. Cette étude nous permettra de compléter nos réflexions sur les outils théoriques, et de les synthétiser sous forme d'une proposition de méthodologie détaillée de conception de Serious Games *(p.150)*.

Au final, si l'étude de l'aspect théorique de la création de Serious Games met en lumière peu d'approches de facilitation, elle nous permet d'identifier de nombreuses spécificités de la création de Serious Games par rapport à la création de jeux vidéo de divertissement *(p.152)*.

OUTILS THEORIQUES DE GAME DESIGN

Ce chapitre est consacré à l'analyse d'une sélection « d'outils théoriques » de Game Design. Comme évoqué précédemment *(p.46)*, ces outils sont généralement inventés et utilisés par les Game Designers professionnels. Leurs principaux vecteurs de diffusion sont les revues spécialisées, que ce soit sur papier tel que *Game Developer Magazine* ***(Think Services, 1994-2011)*** ou en ligne comme *Gamasutra* ***(Think Services, 1997-2011)***, ainsi que par de nombreux ouvrages consacrés au « Game Design ». Afin d'analyser et présenter ces outils théoriques, nous avons opté pour une méthodologie très simple. Nous avons tout d'abord constitué un large corpus de « textes » *(ouvrages, articles de revues…)* consacrés au Game Design. Nous les avons ensuite étudiés, à la recherche de modèles formels faisant office d'outils théoriques destinés à faciliter la création d'un jeu vidéo.

1 ETUDE D'UN CORPUS DE TEXTES TRAITANT DE GAME DESIGN

Selon ***Albinet*** (2010), la formalisation du processus de Game Design à travers des manuels et autres types de textes est relativement récente par rapport à l'histoire du jeu vidéo. Si l'histoire commerciale du jeu vidéo remonte aux débuts des années 1970, ***Albinet*** relève l'émergence du « Game Design » comme thème de théorisation à partir des années 2000. Nous avons donc constitué un **corpus de 40 textes** abordant le Game Design, principalement issus de la période 1999-2010. Le choix de ces textes a été tout simplement effectué par le biais de recherches bibliographiques sur les mots clés *« Game Design »* au sein des ouvrages destinés aux professionnels de l'industrie du jeu vidéo, ainsi que dans les publications académiques. A une exception près, les textes constituants notre corpus sont tous écrits en langue anglaise. 34 d'entre eux sont écrits par des professionnels de l'industrie, alors que seulement 6 sont des publications académiques *(ces dernières sont notées sur fond gris dans le tableau ci-dessous)*.

Tableau 2. Détail du corpus de 40 textes traitant de « Game Design »

Auteur(s)	Titre	Date
Adams	Dogma 2001: A Challenge to Game Designers	2001
Adams	Fundamentals of Game Design	2009
Albinet	Concevoir un jeu vidéo : tout ce que vous devez savoir pour élaborer un jeu vidéo	2010
Bartle	Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs	1996
Bateman & Boon	21st Century Game Design	2005
Bates	Game Design	2004
Bjork & Holopainen	Patterns in Game Design	2004
Brathwaite & Schreiber	Challenges for Game Designers	2008
Bura	A Game Grammar	2006
Church	Formal Abstract Design Tools	1999
Cook	The Chemistry Of Game Design	2007
Crawford	The Art Of Computer Game Design: Reflections Of A Master Game Designer	1982
Crawford	Chris Crawford on Game Design	2003
Dormans	Visualizing Game Mechanics and Emergent Gameplay	2008
Dormans	Machinations: Elemental Feedback Structures for Game Design	2009
Elverdam & Aarseth	Game Classification and Game Design: Construction Through Critical Analysis	2007

Falstein	The 400 Project	2006
Frasca	Simulation versus Narrative: Introduction to Ludology	2003
Fullerton	Game Design Workshop: A Playcentric Approach to Creating Innovative Games	2008
Hunicke & al.	MDA: A Formal Approach to Game Design and Game Research	2004
Järvinen	Games without Frontiers: Theories and Methods for Game Studies and Design	2008
Koster	A Grammar of Gameplay	2005
Koster	Games Are Math: 10 Core Mechanics That Drive Compelling Gameplay	2009
Kremers	Level Design: Concept, Theory, and Practice	2009
Leblanc	Tools for creating Dramatic Games Dynamics	2005
Lecky-Thompson	Video Game Design Revealed	2007
Meigs	Ultimate Game Design: Building Game Worlds	2003
Pardew & al.	Game Design for Teens	2004
Pedersen	Game Design Foundations	2003
Perry	David Perry on Game Design: A Brainstorming Toolbox	2009
Rollings & Adams	Andrew Rollings and Ernest Adams on Game Design	2003
Rollings & Morris	Game Architecture and Design: A New Edition.	2003
Rogers	Level Up!: The Guide to Great Video Game Design	2010
Rouse	Game Design: Theory and Practice	2001
Salen & Zimmerman	Rules of play	2003
Saltzman	Game Design: Secrets of the Sages	1999
Schell	The Art of Game Design: A Deck of Lenses	2008
Schell	The Art of Game Design: A Book of Lenses	2008
Tajè	Gameplay Deconstruction: Elements and Layers	2007
Trefry	Casual Game Design: Designing Play for the Gamer in ALL of Us	2010

L'analyse de ce corpus de textes traitant de « Game Design » nous permet de distinguer deux catégories d'outils théoriques destinés à faciliter la création vidéoludique : les **méthodologies de conception** et les **outils d'aide à la réflexion créative**.

2 LES METHODOLOGIES DE CONCEPTION DE JEUX VIDEO DE DIVERTISSEMENT

Dans le chapitre précédent *(p.54)*, nous avons défini le *« Game Design »* comme le processus global de création d'un jeu vidéo. Une des définitions de la notion de *« processus »* est :

> *« Une série d'étapes permettant d'aboutir à un résultat »*[24]

Ainsi, d'après cette définition, le « résultat » du processus de Game Design est un « jeu ». Nous en avons déjà abordé la définition dans le chapitre précédent *(p.42)*. Nous pouvons à présent nous interroger sur la nature de la « série d'étapes » permettant de créer un tel objet. En d'autres termes, **existe-t-il une « série d'étapes » universelle permettant de concevoir un jeu ?**

2.1 LES MODELES FORMELS DU « PROCESSUS DE CONCEPTION »

Pour tenter de répondre à cette question, nous proposons d'analyser des modèles formalisant le processus de conception d'un jeu que nous avons pu identifier au sein des textes de notre corpus. Plus précisément, six textes de notre corpus proposent une ou plusieurs méthodologies de conception d'un jeu vidéo. Ces diverses méthodologies formalisent tout simplement une « série d'étapes » permettant de créer un « jeu », et définissent donc de manière précise le processus de *« Game Design »*.

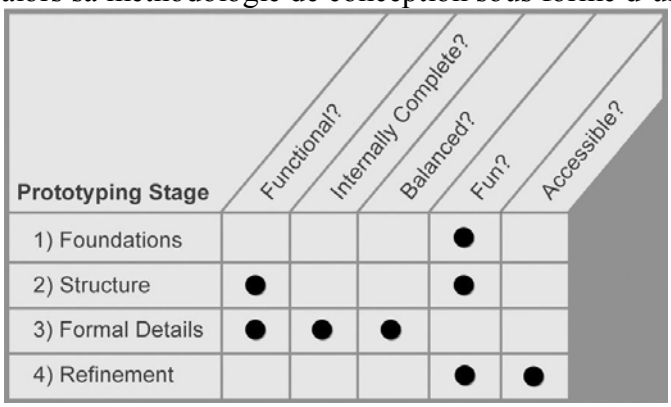
[24] *"A series of events to produce a result."* - Relevé le 20-05-2010 sur http://en.wiktionary.org/wiki/process

2.1.1 LA METHODOLOGIE DE FULLERTON

Un premier modèle formel du processus de « Game Design » est proposé par ***Fullerton*** (2008). Il est basé sur quatre grandes étapes, divisées en de nombreuses sous-étapes :

- *« Fondation »*, étape dans laquelle naît la base du jeu. Elle est divisée en deux grandes parties. Tout d'abord, *« Conceptualisation »*, dédiée à formaliser l'idée de jeu proprement dite. Ensuite vient le *« Prototypage »*, étape qui permet de tester et d'améliorer cette idée par la réalisation de « prototypes », autrement dit des ébauches d'une partie du jeu.
- *« Structure »*, cette étape consiste à utiliser le principe du *« Prototypage »* pour réaliser non plus une partie du jeu, mais une structure complète et fonctionnelle. Pour cela, il faut avoir recours à une autre sous-étape, *« Playtesting »*. Comme son nom l'indique il s'agit de construire le jeu en le faisant tester par des joueurs. Cette démarche doit obéir à un processus itératif : le concepteur crée une première version du jeu, il la fait tester, puis modifie son jeu, le refait tester, et ainsi de suite.
- *« Détails formels »*, une fois la structure du jeu construite, il faut maintenant garnir cette structure en réalisant une version « terminée » du jeu. Pour cela, en utilisant la méthode du *« Playtesting »*, l'auteur invite le concepteur à s'assurer que son jeu remplit trois critères :
  - *« Fonctionnel »*, autrement dit que le jeu est parfaitement utilisable.
  - *« Complet »*, c'est-à-dire que tous les aspects du jeu sont présents.
  - *« Equilibré »*, qui implique que la complexité et la difficulté du jeu sont conformes à l'idée de départ.
- *« Affinage »*, une fois l'intégralité du jeu réalisé, vient l'étape des finitions. Pour cela, ce modèle invite à considérer deux aspects
  - *« Fun »*, pour peaufiner le jeu de manière à ce qu'il soit intéressant pour le joueur.
  - *« Accessibilité »*, pour s'assurer que le jeu est facile à prendre en main.

Fullerton synthétise alors sa méthodologie de conception sous forme d'un tableau :



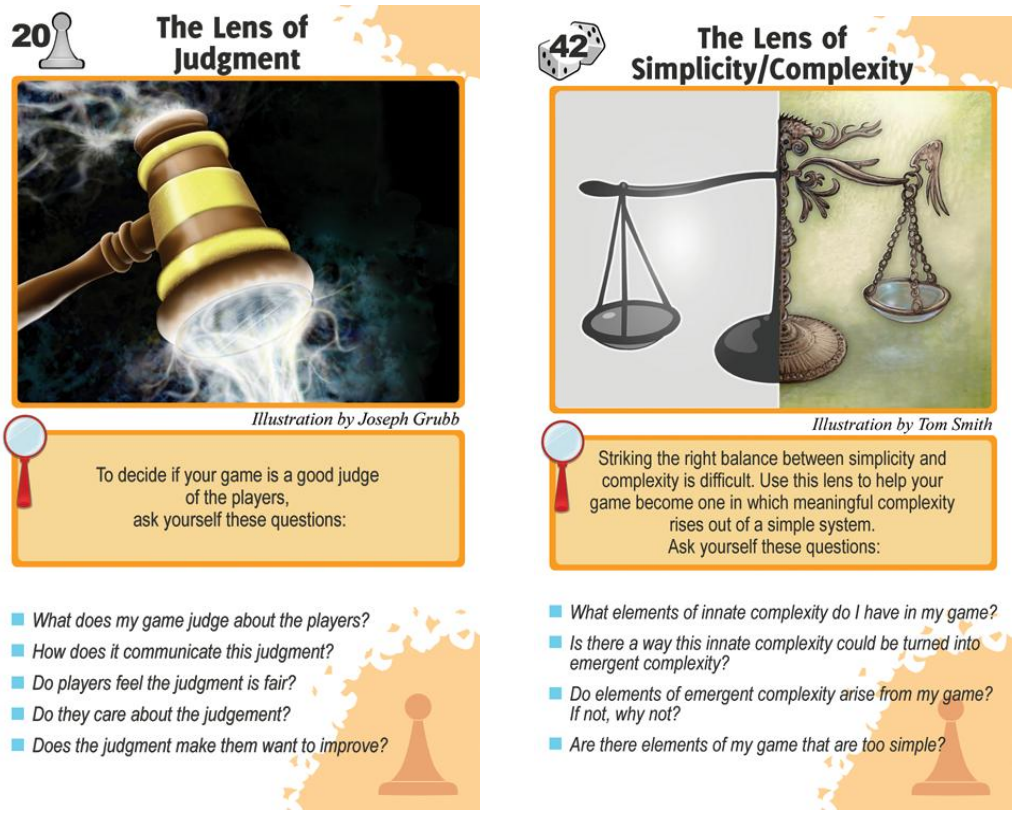
11. Les différentes étapes du processus de conception et les critères de qualité de *Fullerton*

2.1.2 LA METHODOLOGIE DE SCHELL

Dans son ouvrage, ***Schell*** (2008b) propose un modèle de la relation « joueur-jeu » très complet, qui intègre le processus de Game Design *(p.78)*. En plus de ce modèle structuré, le designer propose cent *Lenses*, qui sont des niveaux de lecture portant sur un ou plusieurs des cinq éléments principaux définis dans son modèle : le *« Designer »*, le *« Processus »*, le *« Jeu »*, le *« Joueur »* et *« l'Expérience »*. Ces *Lenses* sont disponibles sous formes de cartes (Schell, 2008a). Ces cartes sont accompagnées d'un manuel décrivant de manière très simple les différentes étapes du processus de Game Design :

- *« Imaginer une idée de jeu. »*
- *« L'essayer. »*
- *« Identifier ce qui ne va pas, le modifier, puis revenir à l'étape précédente. »*

Schell expose ici un processus itératif similaire à celui décrit par ***Fullerton***. Cependant, pour la troisième étape de ce processus, ***Schell*** propose d'utiliser ses *Lenses* pour identifier d'éventuels problèmes. En effet, les *Lenses* sont composées de *« questions à se poser »* pour identifier les éventuels défauts d'un jeu. Chaque *Lens* traite d'un point précis relatif à la création d'un jeu, tel que la manière dont le jeu évalue le joueur ou le niveau de complexité du jeu. Les cartes de ces deux *Lenses* sont présentées ci-dessous :



12. Deux des *Lenses* de *Schell* sous forme de carte

2.1.3 LES METHODOLOGIES SYNTHETIQUES DE BATEMAN & BOON

Une autre approche du processus de conception est proposée par ***Bateman & Boon*** (2005). Elle découle de la notion de *Zen Game Design*, qui est régit par deux principes :

- *« Il n'existe pas de méthode unique pour la conception de jeu. »*
- *« La conception de jeu doit être le reflet de besoins. »*

Si ces deux principes ne formalisent pas les étapes du processus de Game Design à proprement parler, afin d'argumenter leur premier point, les deux concepteurs recensent les étapes de sept processus de Game Design différents :

- Le modèle *« Principe de base »* comprend les étapes *« Définition des objectifs de jeu => Abstraction de l'univers de jeu => Rédaction du document de game design*[25] *=> Réalisation du jeu »*.
- Le modèle *« Copie et Amélioration »* comprend les étapes *« Utilisation d'un Design existant => Modification de ce Design => Réalisation du jeu »*.
- Le modèle *« Méta-Règles »* s'appuie sur les étapes *« Utilisation de Méta-Règles => Rédaction du document de game design => Réalisation du jeu »*. Les « méta-règles » auxquelles il est ici fait référence correspondent aux « pièces à assembler » présentées dans la section suivante *(p.79)*.
- Le modèle *« Expression par la technologie »* se résume à deux étapes *« Création d'une technologie => Réalisation du jeu »*.

[25] Sorte de « cahier des charges » rédigé par le Game Designer, tel qu'expliqué en page 44 de ce mémoire

- *« L'approche Frankenstein »* prône le recyclage de concepts abandonnés : *« Utilisation d'un matériel de départ provenant de projets abandonnés => Rédaction du document de game design => Réalisation du jeu »*.
- Le modèle *« Conception par la narration »* correspond aux étapes *« Ecriture d'une histoire => Rédaction du document de game design => Réalisation du jeu »*
- Enfin, le modèle *« Conception par Itération »* est le seul à introduire des étapes cycliques *« Réunions entre les membres de l'équipe <=> Rédaction du document de game design => Réalisation du jeu »*. Il correspond globalement au processus itératif décrit dans les deux travaux précédents.

Les auteurs précisent qu'il s'agit là d'exemples volontairement simplifiés, mais que ces sept modèles correspondent aux principales méthodologies de Game Design qu'ils ont pu observer durant leur carrière au sein de l'industrie du jeu vidéo. Par extension, leur ouvrage propose même une huitième approche à travers la notion de *Zen Game Design* que nous pourrions retranscrire de la manière suivante : *« Prise en compte du besoin des joueurs grâce au modèle DGD1 => Rédaction du document de game design => Réalisation du jeu »*. Le *modèle DGD1* est un outil d'aide à la réflexion qui formalise des profils de joueurs, tels qu'exposé ultérieurement *(p.72)*.

2.1.4 L'APPROCHE DE ADAMS

Après lecture des diverses approches du Game Design proposées par ***Bateman & Boon***, ***Adams*** (2009) en propose une qui vise à englober toutes ces variantes. Sa formalisation du processus de conception d'un jeu repose sur trois étapes :

- *« La phase de conception »*, dans laquelle le concepteur a une idée et définit des principes généraux qui ne seront pas amenés à changer pendant le reste du processus.
- *« La phase d'élaboration »*, qui suit une logique itérative comme cela a déjà été proposé par d'autres modèles présentés précédemment. Cette phase est consacrée à la définition de toutes les règles du jeu, ainsi qu'à la création de prototypes.
- « *La phase d'ajustement »*, qui vient une fois que toutes les itérations de la phase précédente ont amené un jeu « complet ». D'une manière analogue au modèle de ***Fullerton*** (2008), cette dernière phase permet au concepteur d'ajuster et corriger les dernières imperfections de son jeu.

Ce modèle se distingue des autres par des transitions entre chaque étape clairement identifiées, et par le fait qu'une seule des trois étapes proposées obéit à une logique itérative, tel qu'illustré par le schéma ci-dessous :

13. Le processus de conception vu par *Adams*

2.1.5 L'APPROCHE PIONNIERE DE CRAWFORD

Bien que relativement ancien par rapport au reste du corpus, le premier ouvrage de ***Crawford*** (1982) propose également une série d'étapes composant le processus de Game Design :

- *« Définir un thème et un objectif »*, phase dans laquelle le concepteur réfléchit aux émotions qu'il cherche à procurer aux joueurs tout en définissant un sujet pour le jeu.

- *« Recherche et préparation »*, qui permet au concepteur de se documenter sur le sujet dont va traiter son futur jeu.
- *« Phase de conception »*, après la préparation, la phase de conception peut démarrer. Elle consiste à créer trois aspects du jeu :
    - *« Interfaces entrantes et sortantes »*, les moyens de communication entre le joueur et le jeu.
    - *« Structure du jeu »*, les règles du jeu à proprement parler.
    - *« Structure du programme »*, l'architecture logicielle qui supporte les deux autres aspects.
    - *« Evaluation du Design »* est une phase d'évaluation de la qualité et de la cohérence des trois aspects créés durant la conception. Si l'évaluation n'est pas satisfaisante, il faut abandonner le projet à ce stade, ou recommencer le processus de conception avant d'éventuellement passer à l'étape suivante.
- *« Phase de pré-programmation »*, phase dans laquelle est rédigée le « document de Game Design », sorte de cahier des charges pour la fabrication du jeu *(p.44)*.
- *« Phase de programmation »*, réalisation technique du jeu sur support informatique par le biais d'un langage de programmation.
- *« Phase de tests »*, confrontation du jeu à des joueurs pour en déceler les éventuels derniers défauts.
- *« Post-Mortem »*, où il est important de savoir écouter les critiques des joueurs et de la presse spécialisée car elles peuvent être bénéfiques pour la création d'un prochain jeu.

Notons que ce modèle de conception est intimement lié au support informatique puisqu'il intègre des phases de programmation. Cela s'explique par le fait que cet ouvrage traite uniquement de la conception de jeu vidéo, ainsi que par l'âge du texte, qui représente une des premières tentatives de théorisation du Game Design. Précisons également que l'auteur, bien qu'il consigne sa vision du processus de conception, indique qu'il ne s'agit pas d'une marche à suivre de manière dogmatique. A ses yeux, le processus de conception du jeu reste d'ordre artistique et ne peut être véritablement formalisé vu qu'il dépend aussi de la personnalité de chaque créateur *(p.65)*. Et nous constatons effectivement que les autres ouvrages de ***Crawford*** sur le Game Design ne proposent plus de modèle formel du processus de conception d'un jeu (Crawford, 2003).

2.1.6 UN MODELE GENERAL POUR LA VULGARISATION

Le fait de proposer un modèle représentant le processus « complet » de création d'un jeu, de sa conception à sa réalisation, n'est pas l'apanage de ***Crawford***. Dans leur ouvrage de vulgarisation du Game Design, ***Pardew & al.*** (2004) proposent un modèle intégrant même la phase de distribution du jeu. Ce modèle est articulé autour de quatre étapes principales, divisées en plusieurs sous-étapes :

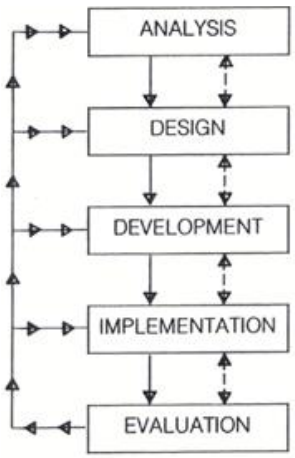
- *« Conception »*, qui possède pour seule sous-étape le fait d'élaborer une *« Idée »*.
- *« Pré-production »*, incluant l'étape de *« Conception »*, de réalisation de « *Prototypes* », jusqu'à une *« Première Version Jouable »*.
- *« Production »*, qui comprend les différentes étapes utilisés dans l'industrie du jeu vidéo pour qualifier la réalisation technique d'un jeu : *« Pre-Alpha »*, *« Alpha »*, *« Beta »* et *« Final »*. Ces étapes correspondent à des versions de plus en plus complètes du jeu.
- *« Produit »*, étape qui distingue ce modèle des autres par l'identification des phases survenant une fois le jeu réalisé : *« Mise en boite »*, *« Campagne Marketing »*, *« Distribution »*, *« Achat par le consommateur »*.

Bien que s'écartant du seul processus de conception, ce modèle a le mérite de brosser le parcours complet de la création d'un jeu vidéo commercial. L'ouvrage précise aussi que la phase de *« Production »* s'appuie sur un processus itératif faisant appel à des tests utilisateurs.

Après avoir passé en revue ces différents modèles formalisant le processus de « Game Design », sommes-nous en mesure de répondre à la question posée en début de chapitre : *existe-t-il une « série d'étapes » universelle permettant de concevoir un jeu ?*

Au premier abord, la variété des modèles que nous venons d'étudier nous inviterait à répondre par la négative : les « séries d'étapes » de ces douze modèles ne sont pas identiques. Une lecture plus poussée nous permet néanmoins d'être plus nuancé. Si en effet les textes de notre corpus ne permettent pas d'identifier une « série d'étapes » universelle pour la conception d'un jeu, certains aspects et étapes semblent récurrents dans les modèles présentés. Tout d'abord, ces douze modèles mentionnent plus ou moins explicitement le caractère itératif de tout ou partie du processus. La notion de prototypage revient également plusieurs fois, de même que la notion de tests auprès des joueurs lors de la conception. Si elle est moins systématique, l'étape de « l'affinage » qui consiste à ajuster un jeu considéré comme « terminé » semble également partagée par plusieurs Game Designers. Ces observations nous poussent à comparer de manière plus structurée les douze modèles issus des six textes étudiés dans cette section. S'il n'est pas possible de définir une méthodologie unique pour la création vidéoludique, peut-être est-il envisageable, en comparant ces modèles, d'identifier des récurrences entre tous les modèles ? Peut-être est-il même possible de formaliser ces récurrences au sein d'un « modèle générique » ?

La notion de « modèle générique » nous est inspirée par un outil issu du secteur de l'ingénierie pédagogique : le *modèle générique ADDIE* (Molenda, 2003). Il s'agit d'un cadre théorique utilisé pour la mise en place de cursus pédagogiques. Ce modèle est composé de cinq étapes : *« Analysis »*, *« Design »*, *« Development »*, *« Implementation »*, et *« Evaluation »*. La phase *« Analysis »* propose au pédagogue d'identifier l'objectif pédagogique visé, ainsi que d'analyser le public ciblé. La phase *« Design »* permet de concevoir les différentes étapes du cursus pédagogique. La phase *« Development »* incite alors à créer tous les contenus pédagogiques du cursus. Ce cursus sera testé auprès d'un public témoin *(enseignants et élèves)* lors de phase *« Implementation »*. Enfin, la phase *« Evaluation »* propose d'évaluer la pertinence du cursus selon deux méthodes : l'évaluation *« formative »*, qui est continue tout au long du processus de création, et l'évaluation *« sommative »* qui vise à tester certains aspects du cursus auprès d'un public spécifique.



14. Représentation schématique du *modèle générique ADDIE*

Si ce modèle est considéré comme « générique », c'est parce que les cinq étapes qu'il définit sont rarement utilisées telles qu'elles. Généralement, chaque pédagogue définit sa propre « série d'étape » en s'appuyant sur le cadre proposé par ce modèle. Comme l'explique ***Molenda*** dans son article, cela est visiblement imputable au fait que l'appellation *ADDIE* n'a

pas d'origine précise et identifiée, mais est le fruit d'une pratique empirique du secteur de l'ingénierie pédagogique. Chaque pédagogue est ainsi libre de proposer son propre modèle d'ingénierie pédagogique en inventant un nombre variable d'étapes qui s'inscrivent au sein des cinq « étapes génériques » définies par le *modèle générique ADDIE*.

Cette approche de « modèle générique » nous semble particulièrement pertinente pour comparer les méthodologies de « Game Design » que nous avons étudiés. Nous avons donc comparé les différentes étapes de ces douze modèles, et avons essayé de les regrouper de manière thématique. Par exemple, nous remarquons que de nombreuses étapes des douze modèles étudiés consistent globalement à imaginer un concept de jeu : ***Adams*** parle de *« phase de conception »*, ***Fullerton*** de *« conceptualisation »* tandis que ***Schell*** précise que la première étape consiste à *« imaginer une idée de jeu »*. De même, les trois premières étapes d'un des modèles ***Bateman & Boon*** semblent détailler les étapes permettant d'imaginer une concept de jeu : *« définition des objectifs de jeu »*, *« abstraction de l'univers de jeu »*, *« rédaction du document de game design »*. Nous proposons alors de regrouper ces étapes de nos différents modèles au sein d'une « étape générique » tout simplement baptisée *« Imaginer »*. En suivant une démarche similaire pour l'ensemble des étapes formalisées dans nous douze modèles, nous obtenons un total de « trois étapes génériques » :

- **Imaginer :** le concepteur invente un concept de jeu. Cette étape va généralement de pair avec l'emploi d'outils théoriques.
- **Créer :** un prototype est réalisé pour tester la pertinence de ce concept de jeu. Cette étape est généralement appuyée par l'utilisation d'outils techniques.
- **Evaluer :** le prototype est évalué auprès d'un public cible. Les critères d'évaluation varient selon les projets, mais dans le cas d'un jeu de divertissement il s'agira généralement d'évaluer si les joueurs s'amusent avec le jeu *(notion de « fun », p.73)*.

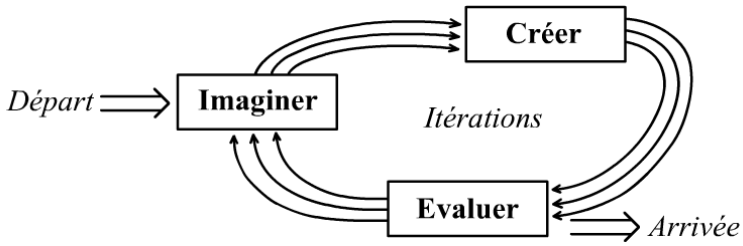
Le tableau ci-dessous expose la manière dont ces trois « étapes génériques » permettent de regrouper les séries d'étapes des douze modèles du processus de conception de jeux vidéo étudiés dans ce chapitre :

Tableau 3. Comparaison des modèles du processus de Game Design à travers le *modèle générique ICE*

Outil théorique	Imaginer	Créer	Evaluer
Modèle de ***Fullerton*** *(p.61)*	- Fondation (Conceptualisation)	- Fondation (Prototypage) - Structure (Prototypage) - Détails formels (Réalisation)	- Structure (Playtesting) - Détails formels (Playtesting) - Affinage
Modèle de ***Schell*** *(p.61)*	- Imaginer une idée de jeu	- L'essayer - …le modifier	- Identifier ce qui ne va pas,…
Modèle #1 de ***Bateman & Boon*** *(p.62)*	- Définition des objectifs de jeu - Abstraction de l'univers de jeu - Rédaction du document de game design	- Réalisation du jeu	-
Modèle #2 de ***Bateman & Boon*** *(p.62)*	- Utilisation d'un design existant - Modification de ce design	- Réalisation du jeu	-
Modèle #3 de ***Bateman & Boon*** *(p.62)*	- Utilisation de méta-règles - Rédaction du document de game design	- Réalisation du jeu	-

Modèle #4 de ***Bateman & Boon*** *(p.62)*	-	- Création d'une technologie - Réalisation du jeu	-
Modèle #5 de ***Bateman & Boon*** *(p.62)*	- Utilisation d'un matériel de départ provenant de projets abandonnés - Rédaction du document de game design	- Réalisation du jeu	-
Modèle #6 de ***Bateman & Boon*** *(p.62)*	- Ecriture d'une histoire - Rédaction du document de game design	- Réalisation du jeu	-
Modèle #7 de ***Bateman & Boon*** *(p.62)*	- Réunions entre les membres de l'équipe - Rédaction du document de game design	- Réalisation du jeu	-
Modèle de ***Adams*** *(p.63)*	- Phase de conception	- Phase d'élaboration	- Phase d'ajustement
Modèle de ***Crawford*** *(p.63)*	- Définir un thème et un objectif - Recherche et préparation - Phase de conception	- Phase de pré-programmation - Phase de programmation	- Phase de tests - Post-Mortem
Modèle de ***Pardew & al.*** *(p.64)*	- Conception	- Préproduction - Production	- Produit

A défaut d'identifier une série d'étape universelle du processus de Game Design, nous pouvons alors envisager de définir un « modèle générique » fonctionnant comme le *modèle générique ADDIE* : une proposition générique destinée à inspirer les Game Designers pour définir leur propre série d'étapes. Ce « modèle générique » reposera sur les trois « étapes génériques » que nous venons de définir à partir de l'analyse de douze méthodologies de conception vidéoludique : *Imaginer*, *Créer*, *Evaluer*. Nous appellerons donc ce modèle, le *modèle générique ICE*. Un autre aspect récurrent dans les modèles étudiés, mais qui n'apparaît pas dans cette série d'étapes, est la notion de cycles itératifs. Nous proposons alors de définir le *modèle générique ICE* comme reposant intégralement sur un cycle itératif. Ce cycle commence par une étape *« Imaginer »* pour s'achever après une étape *« Evaluer »*, tel que représenté par le schéma ci-dessous :



15. Le *modèle générique ICE* du processus de Game Design

Volontairement très simple, le *modèle générique ICE* n'est pas sans évoquer les méthodologies de conception et de gestion de projet provenant d'autres domaines, par exemple les méthodes agiles de développement informatique dites « en spirale ». D'un point de vue méthodologique, la création de jeux vidéo ne semble donc pas très éloignée de la réalisation d'autres types de programmes informatiques.

3 LES OUTILS D'AIDE A LA REFLEXION CREATIVE

Au-delà de la proposition d'une « série d'étapes » permettant la création d'un jeu vidéo, les textes que nous avons étudiés proposent de nombreux autres outils théoriques destinés à guider le concepteur dans sa réflexion créative. Par exemple, certains outils proposent des méthodes de modélisation graphique de la structure d'un jeu *(p.68)*, afin d'aider le concepteur à inventer des concepts de jeux cohérents. D'autres proposent encore d'analyser les joueurs *(p.71)* et la manière dont ils interagissent avec un jeu *(p.73)*, toujours afin d'aider le concepteur à créer un jeu pertinent. D'autres textes proposent enfin des « pièces à assembler » permettant de créer un jeu plus rapidement *(p.79)*.

Tous ces outils théoriques visent à guider le créateur d'un jeu vidéo dans sa démarche de réflexion créative. En regard du *modèle générique ICE* que nous venons de définir, ces outils sont donc destinés à être utilisé durant l'étape *« Imaginer »*.

3.1 LES MODELES FORMELS DU « JEU »

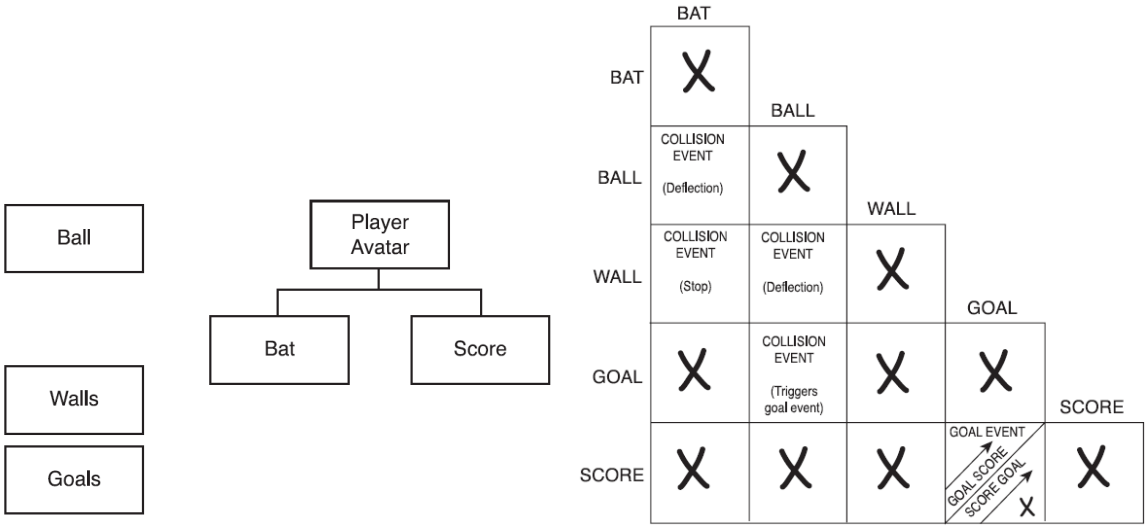
Cette catégorie d'outils théorique rassemble les huit textes de notre corpus qui proposent un modèle formel du « jeu » en tant qu'objet, visant à détailler sa structure interne.

3.1.1 LES MODELES TYPOLOGIQUES

Nos deux premiers modèles sont d'ordre « typologique » : le modèle met en évidence un nombre fini d'éléments composant la structure d'un jeu, et peut donc être utilisé comme un « plan » permettant d'en créer de nouveaux. La *typologie multidimensionnelle des jeux* proposée par ***Elverdam & Aarseth*** (2007) en est le parfait exemple. Elle détaille de manière très précise 16 aspects différents d'un jeu allant de sa représentation de l'espace *(réel ou virtuel)* à la manière de gérer l'état courant du jeu, en passant par la présence d'objectifs ou le nombre de joueurs. Si ce modèle essaie de traiter tous les aspects d'un jeu, la *typologie des règles* introduite par ***Frasca*** (2003) se focalise uniquement sur les règles de jeu. Elle en met en évidence trois types : les *« règles de manipulation »*, les *« règles d'objectifs »* et les *« méta-règles »*. La portée limitée de ces deux modèles, qui se contentent de « lister » les différents types d'éléments constitutif d'un jeu, permet d'envisager les utiliser avec des modèles plus élaborés tels que ceux présentés ci-après.

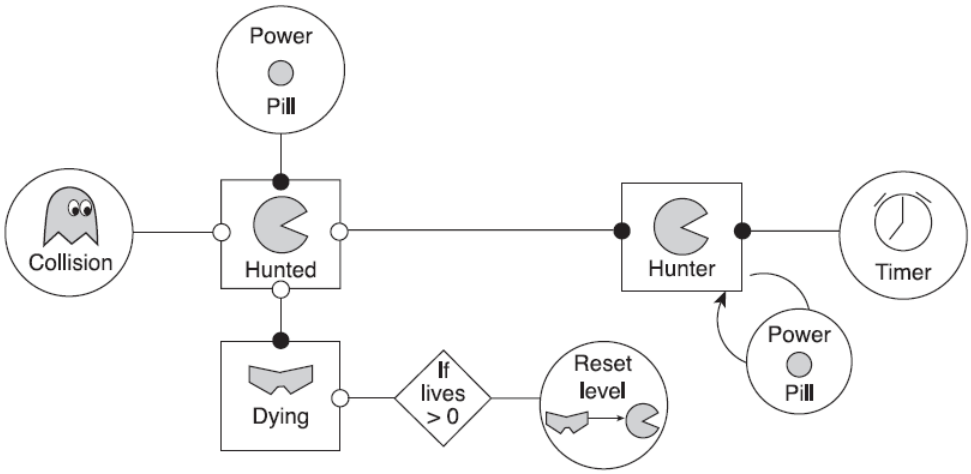
3.1.2 LA THEORIE DES « TOKENS »

La *théorie des Tokens* est introduite par ***Rollings & Morris*** (2003). Elle propose de diviser les différents éléments d'un jeu en petits éléments unitaires baptisés *« tokens »*. Un *« token »* est tout simplement un élément conceptuel du jeu. Un *« token »* peut contenir d'autres *« token »*, permettant donc de formaliser une représentation hiérarchique des différents éléments du jeu. Dans leur ouvrage, les deux auteurs proposent une représentation de *Pong* ***(Atari, 1972)*** en cinq *« tokens »* : la balle, les raquettes, les murs, les « buts » *(zones où l'on doit envoyer la balle pour marquer)* et les compteurs de score. Les raquettes et les compteurs de score sont deux *« tokens »* qui font partie d'un *« token »* plus large représentant le joueur. A partir de cette hiérarchie, il est possible de noter facilement les interactions entre les différents *« tokens »*. Pour continuer avec l'exemple de *Pong*, un tableau permet de noter que la raquette interagit avec la balle en la renvoyant, qu'une raquette sera stoppée par un mur ou qu'une collision entre un but et la balle agira sur un des compteurs de score en l'incrémentant. Ce « tableau » est appelé une matrice d'interaction.



16. La hiérarchie de *« tokens »* du jeu *Pong* (gauche) et la matrice d'interaction associée (droite)

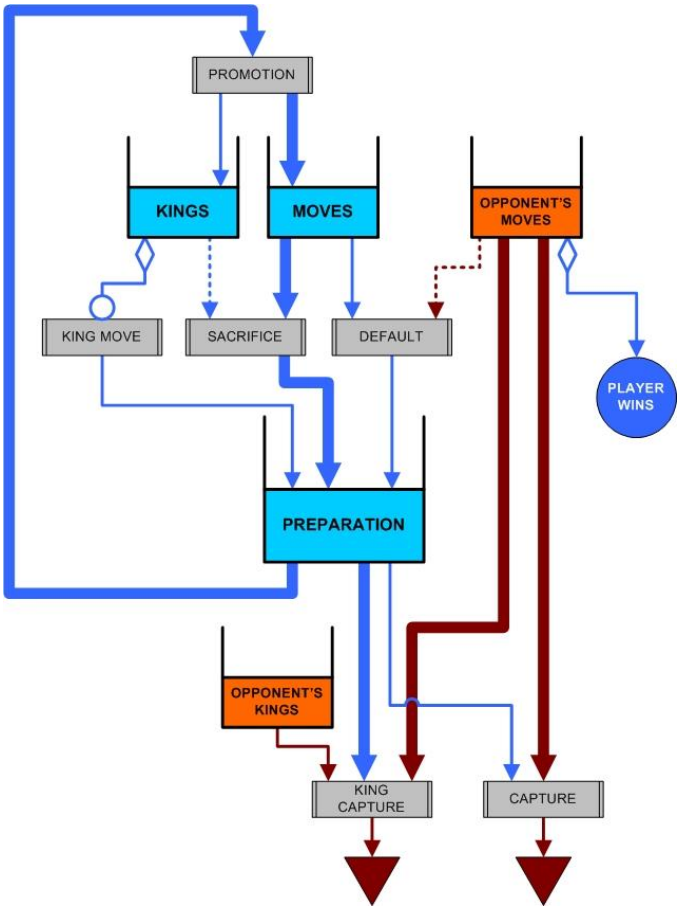
Si ce mode de conception est adapté aux jeux simples, il montre rapidement ses limites. Les auteurs proposent donc une évolution de ce modèle où les interactions ne sont plus notées directement entre *« tokens »*, mais entre les propriétés et états de ces *« tokens »*. Cela permet de concevoir des jeux plus complexes où les *« tokens »* sont eux-mêmes des éléments à état variable. Ils proposent même d'utiliser une représentation graphique séparée sous forme de *« Machine à Etat Fini » (Finite State Machine)* de chacun de ces *« tokens »*, afin de pouvoir clairement lister leurs différents états et les relations entre chacun de ces états. Le schéma ci-dessous illustre ce principe pour un élément du jeu *Pac-Man* ***(Namco, 1980)***.



17. La représentation d'un *« token »* à état variable pour le jeu *Pac-Man*

3.1.3 LES « LUDEMES » ET LA « GAME GRAMMAR »

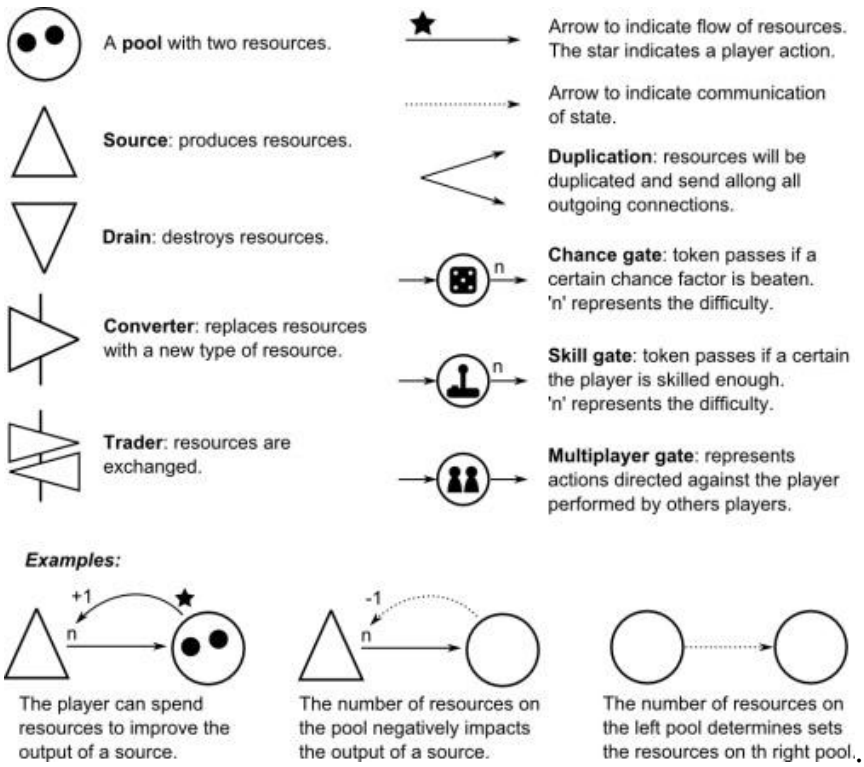
Offrant un principe de fonctionnement proche des *« tokens »*, les *« ludemes »* introduits par ***Koster*** (2005) proposent eux aussi de diviser les jeux en éléments unitaires, et de travailler leur mise en relation pour concevoir l'interactivité ludique. Inspiré par l'approche de ***Koster***, ***Bura*** (2006) propose un modèle similaire, mais s'appuie sur une représentation graphique inspirée par les *réseaux de Pétri*. Les « jetons » de ce célèbre modèle mathématique sont utilisés pour représenter les ressources dans les règles du jeu, tandis que les « places » représentent les éléments de jeu *(correspondants aux « ludemes » de **Koster**)*, alors que les « transitions » modélisent les actions que peuvent effectuer les joueurs.



18. La représentation du jeu des dames *(checkers)* selon la *Game Grammar* de *Bura*

3.1.4 LES « MACHINATIONS » DÉRIVÉES D'UML

Le fait qu'un modèle s'appuie sur un formalisme existant permet aux concepteurs de jeux de l'appréhender plus facilement. Influencé par ***Bura***, un chercheur du nom de ***Dormans*** s'est inspiré du standard *UML*. En utilisant les « diagrammes de collaboration » de ce langage, il propose un premier modèle permettant de modéliser la structure d'un jeu en respectant scrupuleusement le standard *UML* (Dormans, 2008). Ce modèle a ensuite évolué vers un mode de représentation personnalisé, mais toujours inspiré par *UML* (Dormans, 2009). Baptisé *Machinations*, ce modèle est particulièrement adapté à la conception de jeux basés sur une économie interne, autrement dit les jeux où la notion de « ressources » est fondamentale. Le principe de représentation graphique consiste tout d'abord à définir des « réservoirs de ressources » qui représentent les éléments du jeu. Ensuite, ces différents réservoirs peuvent interagir entre eux selon quatre modalités : produire des ressources, les détruire, les convertir ou les échanger. Des modalités d'interaction secondaires sont également disponibles, comme le montre le schéma ci-dessous :



19. Listes des éléments de modélisation du modèle *Machinations*

Plus qu'un simple outil théorique, ***Dormans*** a également inventé un logiciel proposant de dessiner des diagrammes utilisant son formalisme (Dormans, 2011). Ce logiciel permet de « tester » le jeu résultant de manière sommaire, en simulant la circulation des ressources de manière automatique.

3.1.5 LE « OBJECT-ORIENTED » GAME DESIGN

S'inspirant également de concepts informatiques, ***Lecky-Thompson*** (2007) introduit la notion de *Object-Oriented Game Design*. Concrètement, l'auteur propose de diviser un jeu non pas en *« tokens »* ou en *« ludemes »*, mais directement en « objets » qui obéissent au paradigme de programmation informatique du même nom (Roy & Haridi, 2004). Le « modèle Objet » est une déclinaison du paradigme de programmation « impératif » dans laquelle les programmes sont composés par des « objets » possédants des propriétés *(valeurs stockées en mémoire)* et des méthodes *(capacité à effectuer des séries de calculs)*. La force de ce modèle vient du fait que les objets peuvent communiquer entre eux, mais également posséder une filiation : certains objets peuvent « hériter » les compétences des autres. Une différence majeure entre ce paradigme et les *« tokens »* vient du fait que les « objets » possèdent à la fois les informations sur l'état du jeu et la capacité d'appliquer les règles, alors que les *« tokens »* et *« ludemes »* sont uniquement utilisés pour mémoriser l'état actuel du jeu. Nous retrouverons d'ailleurs cette différence d'approche avec les outils techniques *(p.179)*. Au final, bien que ce Game Designer ne donne pas d'exemple concret de sa méthode, le fait de s'appuyer sur un modèle formel issu du monde de la programmation informatique est visiblement une approche permettant de simplifier la création vidéoludique pour les personnes familières avec la programmation « Orienté Objet »

3.2 LES MODELES FORMELS DU « JOUEUR »

Cette catégorie rassemble trois textes qui, par le biais d'un modèle formel, tentent de comprendre les joueurs et leurs motivations. Contrairement aux autres catégories, des travaux antérieurs aux années 2000 y sont recensés. L'étude du joueur est un sujet à part entière qui a déjà amené plusieurs travaux dans le champ des sciences humaines. Afin de limiter l'étendue

de notre étude, nous ne présentons ici que des modèles auxquels il est directement fait référence dans les textes constituant notre corpus.

3.2.1 LA TYPOLOGIE DE CAILLOIS

En premier lieu, les travaux de ***Caillois*** (1967), bien que ne faisant pas partie de notre corpus car datant de 1958, sont cités dans plusieurs des ouvrages, notamment ceux de ***Salen & Zimmerman*** (2003), de ***Bateman & Boon*** (2005) ou encore d'***Albinet*** (2010). Plus particulièrement, ces ouvrages s'appuient sur la classification des jeux proposés par le sociologue afin de bâtir un modèle formel du joueur. ***Caillois*** distingue quatre catégories :

- *« Agôn »*, les jeux basés sur la compétition.
- *« Alea »*, les jeux basés sur le hasard.
- *« Mimicry »*, les jeux basés sur le simulacre.
- *« Ilinx »*, les jeux basés sur le vertige.

En classifiant ainsi les jeux selon l'attitude des joueurs qui le pratiquent, les travaux de ***Caillois*** sont utilisés pour distinguer quatre catégories de « joueurs » : ceux qui s'intéressent à la compétition, ceux qui cherchent à défier le hasard, ceux attirés par le simulacre et ceux recherchant la sensation de vertige. Ce modèle peut-être utilisé comme le point de départ de la conception, par exemple en tentant d'imaginer à un jeu s'inscrivant dans une de ces catégories.

3.2.2 LE MODELE « DGD1 » DE BATEMAN & BOON

D'autres modèles du joueur se sont inspirés de l'approche pionnière de ***Caillois*** pour aboutir à leur propre catégorisation. Une des plus abouties semble être le modèle *Demographic Game Design 1 (DGD1)* introduit par ***Bateman & Boon*** (2005). Il s'appuie sur les « types psychologiques » définit par ***Myers-Briggs*** (Briggs-Myers & P. B. Myers, 1980) pour proposer quatre grandes catégories de joueurs :

- *« Conqueror »*, les joueurs assoiffés de victoire et de récompense à leurs efforts.
- *« Manager »*, les joueurs à la recherche d'un défi à la hauteur de leur capacité de réflexion, de stratégie, de tactique ou de gestion.
- *« Wanderer »*, les joueurs qui recherchent avant tout du divertissement immédiat et de l'amusement facile.
- *« Participant »*, les joueurs cherchant les expériences sociales tout en étant sensibles à la dimension narrative.

Ces quatre catégories sont ensuite modulés par un facteur *« hardcore player » (joueur expert)* ou *« casual player » (joueur occasionnel)*, ce qui fait un total de huit catégories. Ces catégories ne sont pas « exclusives » : un même joueur peut se retrouver dans plusieurs catégories à la fois.

3.2.3 LA TYPOLOGIE DE BARTLE

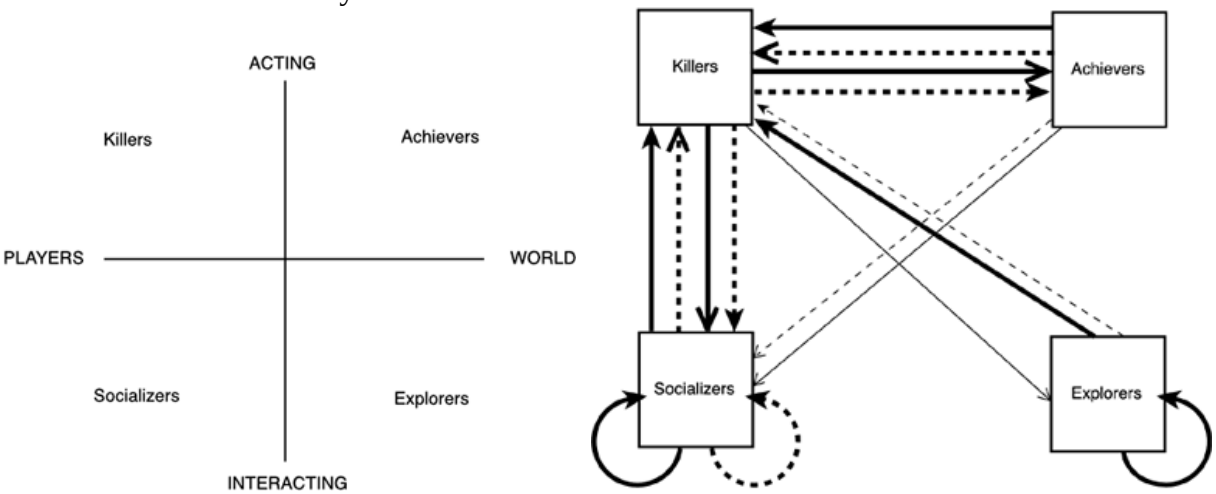
Cette notion de catégories non-exclusives de joueurs est également la base d'une autre typologie des joueurs, celle proposée par ***Bartle*** (2005). Célèbre pour avoir été le co-concepteur d'un des premiers *MUD*[26] en 1978, il a également passé de nombreuses années à étudier les joueurs de ces premiers jeux de rôle en ligne. La typologie des joueurs qu'il a proposée en 1996 définit quatre profils de joueurs pour les jeux multijoueurs :

- *« Achievers »*, les joueurs à la recherche de défis à relever, qui tirent satisfaction de leur montée en puissance dans le jeu.

[26] Acronyme de "**M**ulti **U**ser **D**ungeon", qui désigne des jeux de rôle multijoueurs en réseau. Il s'agit des ancêtres des jeux de rôles massivement multijoueurs tels que *World of Warcraft* ***(Blizzard, 2004)***. Le premier jeu connu du genre, baptisé tout simplement *MUD* a été co-écrit par ***Roy Trubshaw*** et ***Richard Bartle*** en 1978.

- *« Explorers »*, les joueurs guidés par le désir de connaissance, qui passent le plus clair de leur temps à essayer de comprendre le fonctionnement du jeu plutôt qu'à essayer de le « terminer ».
- *« Socializers »*, les joueurs qui viennent avant tout pour rencontrer d'autres joueurs et discuter.
- *« Killers »*, l'anti-thèse des *« Socializers »*, qui jouent dans le seul but de nuire aux autres joueurs.

Ces profils stéréotypés ne sont que des « tendances comportementales » que les joueurs adoptent de manière non exclusive. Les travaux de ***Bartle*** nous apprennent également que le bon fonctionnement d'un jeu multijoueurs est soumis à une répartition équilibrée des quatre types de joueurs. Plus qu'une simple typologie, ce modèle met donc en évidence les relations entre chaque catégorie de joueurs afin de permettre aux concepteurs de jeux multijoueurs d'arriver à créer un « écosystème » durable.



20. Les centres d'intérêt des profils de joueur (gauche) et leurs relations (droite) selon *Bartle*

3.3 LES MODELES FORMELS DE « LA RELATION JOUEUR-JEU »

Les six textes de cette catégorie visent à proposer un modèle formel permettant d'analyser la relation entre le joueur et le jeu. Dans cette optique, certains de ces travaux sont particulièrement complets car ils apportent également un modèle du joueur et/ou du jeu, tout en incluant ce ou ces modèles dans une réflexion plus large.

3.3.1 L'ALCHIMIE DE COOK

Cook (2007) propose un des modèles les plus complets et détaillés du jeu, du joueur, et de leur relation. Le modèle formalisant le joueur est simple, ce dernier étant vu comme :

> *« [U]ne entité qui cherche, consciemment ou inconsciemment, à apprendre de nouvelles compétences qu'il considère importantes. Son plaisir découle directement de l'apprentissage de nouvelles compétences »*[27].

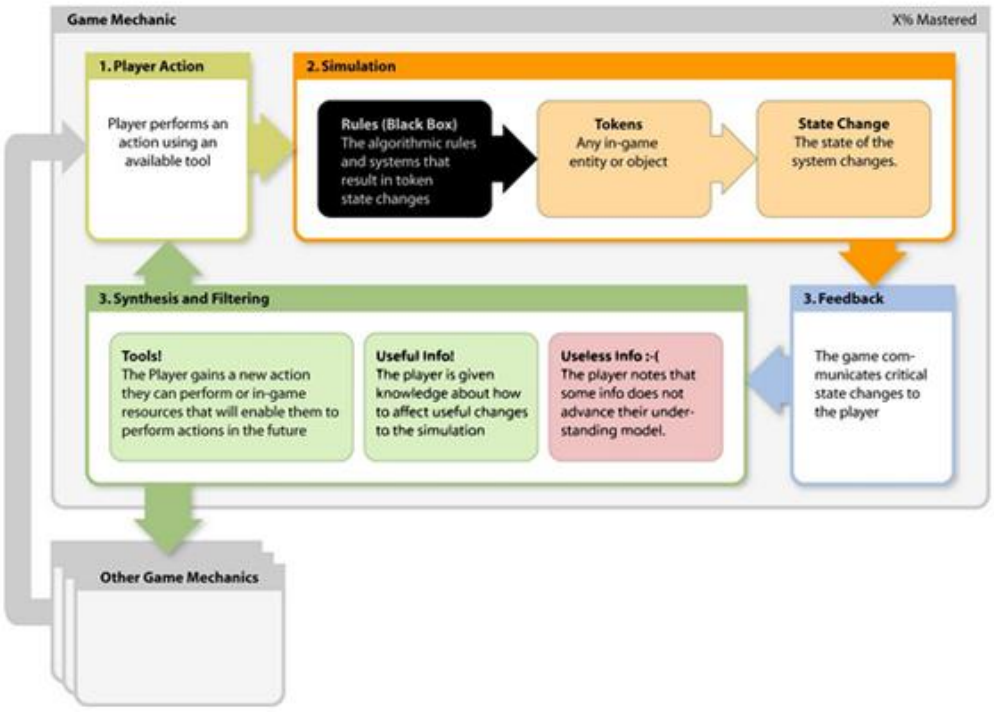
En ce sens, il rejoint les propos de ***Koster*** (2004) et sa théorie des « patterns » qui amène ce dernier à définir le *« fun »* comme :

[27] *"The player is entity that is* ***driven****, consciously or subconsciously, to learn new* ***skills*** *high in* ***perceived value****. They gain pleasure from successfully acquiring skills."*

> *« La façon dont notre cerveau nous récompense lorsque nous assimilons de nouvelles compétences »*[28] *[p.107]*

Cook propose ensuite un modèle formalisant la relation entre le joueur et l'objet jeu. Ce modèle divise un jeu en plusieurs *« mécanismes »*. Chacun de ces *« mécanismes »* sert à l'apprentissage d'une *« compétence »* précise, telle que le fait d'arriver à sauter ou de comprendre que lorsque *Mario* récolte cent pièces il gagne une vie supplémentaire. Dans l'esprit de ***Cook***, une *« compétence »* est donc tout simplement liée à la compréhension du jeu et de son fonctionnement, et non pas forcément à quelque chose d'utile et d'applicable dans la « vie réelle ». Plusieurs *« mécanismes »* peuvent être liés à une même *« compétence »*. Ces *« mécanismes »* divisent la relation « joueur-jeu » en quatre phases :

- *« Actions du joueur »* : le joueur effectue une action.
- *« Simulation »* : en utilisant ses règles, le jeu modifie ses *« tokens »* internes pour changer « d'état » *(à noter ici que **Cook** s'appuie sur la notion de « tokens » exposée précédemment, cette dernière étant apparemment relativement répandue dans l'industrie du jeu vidéo).*
- *« Retour »* : le jeu communique son changement d'état au joueur.
- *« Synthèse et discrimination »* : mentalement, le joueur élimine les informations qui ne lui semblent pas pertinentes pour « comprendre » le jeu, puis assimile les informations qu'il a jugé pertinentes afin d'essayer d'apprendre une nouvelle *« compétence »*, et donc d'avancer dans sa compréhension du jeu.

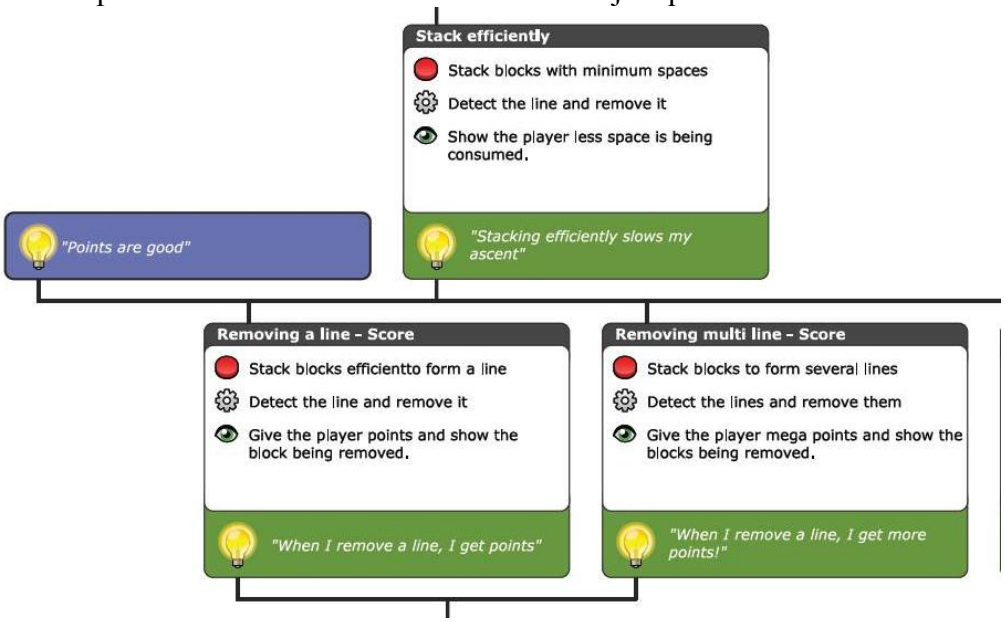


21. Détail d'un mécanisme de jeu selon l'alchimie de *Cook*

Cook utilise ensuite ces *« mécanismes »* comme éléments unitaires pour représenter, sous forme de diagrammes, la relation entre le joueur et le jeu. En analogie à la chimie, chaque *« mécanisme »* est considéré comme un « atome » qui peut-être lié à un ou plusieurs autres *« mécanismes »*, afin de créer une *« chaîne d'apprentissage de compétences »* à la structure non-linéaire. Cette représentation formelle liste toutes les *« compétences »* que le joueur peut retirer d'un jeu donné, et les différentes façons lui permettant de les acquérir. En listant ainsi

[28] *"Fun, as I define it, is the feedback the brains gives us when we are absorbing new patterns for learning purposes"*

les interactions potentielles entre le joueur et le jeu, ***Cook*** propose un outil permettant au concepteur de représenter visuellement l'interactivité du jeu qu'il en train de créer.



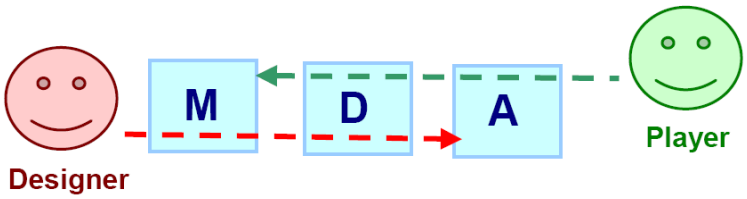
22. Extrait de la représentation de *Tetris* par *Cook*

3.3.2 LE MODELE « MDA »

En plus de ce modèle très précis, d'autres travaux proposent de modéliser la relation « joueur-jeu » avec différents niveaux de précision. Le plus simple est sans doute le *modèle MDA* (Hunicke, Leblanc, & Zubek, 2004), qui divise la relation « joueur-jeu » en trois niveaux de lectures distincts :

- « ***M**echanics* », qui propose d'analyser le jeu au niveau des « règles de jeu ».
- « ***D**ynamics* », qui se focalise sur le système qui naît lorsque le joueur utilise le jeu.
- « ***A**esthetics* », dédié à la compréhension et l'analyse du ressenti du joueur.

Ces trois niveaux d'analyse théorique sont un moyen de positionner le joueur et le concepteur du jeu. Le concepteur du jeu travaille sur les *« Mechanics »*, afin de contrôler indirectement les *« Dynamics »* pour essayer de susciter une réaction donnée de la part du joueur dans les *« Aesthetics »*. A l'inverse, le joueur se base sur son ressenti *(« Aesthetics »)* pour interagir avec le jeu *(« Dynamics »)* dans le but d'en comprendre le fonctionnement profond *(« Mechanics »)*.



23. Positionnements respectifs du concepteur et du joueur selon le *modèle MDA*

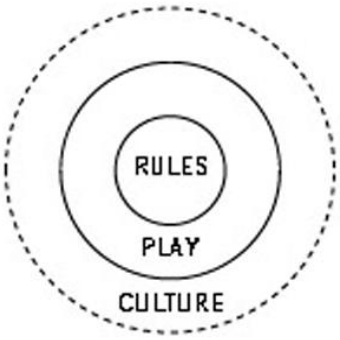
3.3.3 LES « PRIMARY SCHEMAS » DE SALEN & ZIMMERMAN

Un autre modèle de la relation « joueur-jeu » est basé sur une structure tricéphale, celui des *Primary Schemas* introduit par ***Salen & Zimmerman*** (2003). Les auteurs proposent trois *« schémas »*, qui sont des niveaux de lecture sous forme de couches concentriques allant du jeu vers le joueur. Les deux chercheurs les définissent de la manière suivante :

- Le schéma *« Rules »* est d'ordre formel, et permet de se focaliser sur l'analyse de la structure mathématique intrinsèque au jeu.
- Le schéma *« Play »* est d'ordre expérientiel, et se consacre à l'analyse des interactions directes entre le joueur et le jeu ainsi qu'entre les différents joueurs.

- Le schéma *« Culture »* est d'ordre contextuel, et met en évidence le contexte culturel dans lequel tout jeu est amené à être pratiqué.

A la lecture de ces définitions, il est impossible de ne pas imaginer une certaine influence des *Primary Schemas* sur le *modèle MDA*. Si ce dernier semble aujourd'hui plus souvent cité comme référence, il ne faut pas pour autant négliger l'apport de ***Salen & Zimmerman*** sur la prise en compte de trois dimensions lors de l'analyse de la relation « joueur-jeu ». Par exemple, dans sa thèse ***Alvarez*** (2007) analyse le Serious Game par le biais de trois niveaux : *« formel », « pragmatique »* et *« culturel »*. Il s'agit d'une référence directe aux *« schémas primaires »* présentés ici.



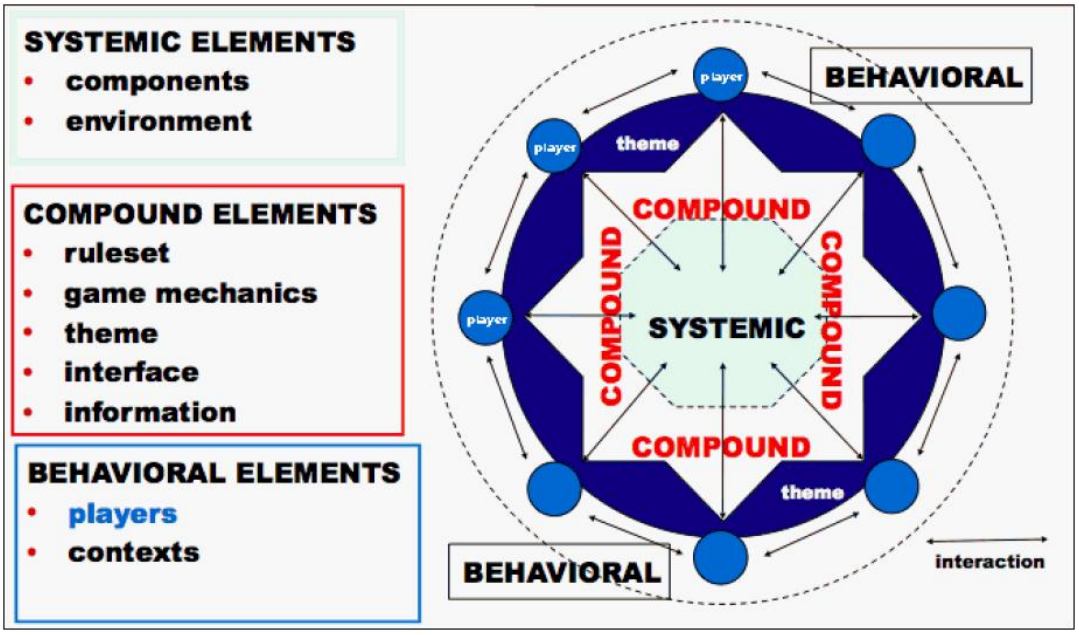
24. Hiérarchie des trois *Primary Schemas* de *Salen & Zimmerman*

3.3.4 LES « GAME ELEMENTS » DE JÄRVINEN

Autre modèle d'analyse à la philosophie similaire au *modèle MDA*, les *Game Elements* de ***Järvinen*** (2008) proposent neuf niveaux de lecture de la relation « joueur-jeu ». Malgré un nombre plus élevé de niveaux de lecture, cet outil est bien une évolution du *modèle MDA*. En effet, ***Järvinen*** regroupe ses neufs éléments en trois catégories qui rappellent celles définies par ses prédécesseurs :

- Les *« éléments systémiques »*, qui rassemblent les *« composants »* et *« l'environnement »* sont une réinterprétation du niveau des *« Dynamics »*.
- Les *« éléments composés »* regroupent les *« règles »*, les *« mécanismes de jeux »*, le *« thème »*, *« l'interface »* et les *« informations »*. En d'autres termes, ce groupe détaille tous les éléments sur lesquels peut agir directement le concepteur du jeu, et renvoie donc à la couche *« Mechanics »*.
- Les *« éléments comportementaux »* se composent des *« joueurs »* et des *« contextes »*, afin de proposer une analyse plus fine du niveau *« Aesthetics »*.

En plus de détailler les différentes composantes de chacune des couches du *modèle MDA*, ***Järvinen*** en modifie également l'organisation. Comme le montre le schéma ci-dessous, la couche *« éléments systémiques » (correspondant aux « Dynamics »)* devient centrale. C'est donc la couche des *« éléments composés » (soit les « Mechanics »)* qui permet au joueur d'accéder à l'interaction ludique, là où le *modèle MDA* voit l'interaction ludique comme le fruit de la rencontre entre les mécanismes et le joueur.



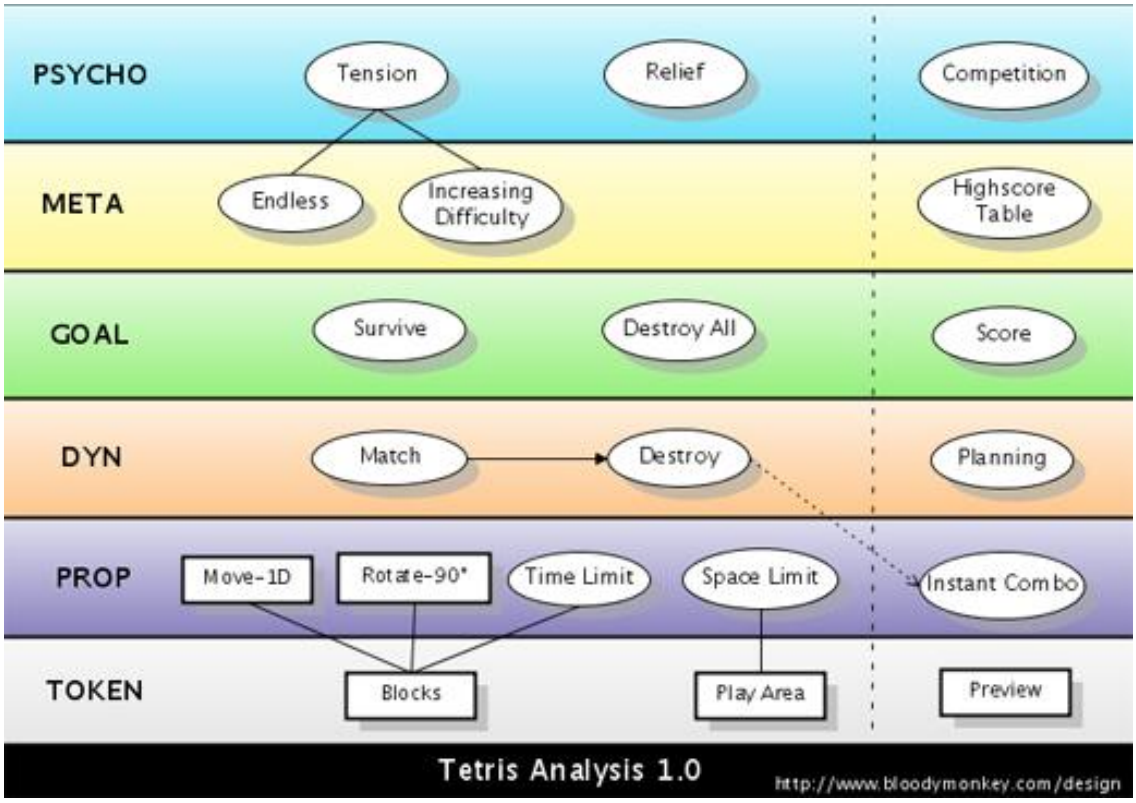
25. Représentation des différents *Game Elements*

3.3.5 LES « GAME LAYERS » DE TAJE

Toujours dans l'optique d'établir des niveaux de lecture, nous retrouvons les travaux de ***Tajè*** (2007) qui propose une approche reposant sur la division de la relation « joueur-jeu » en six couches. Les *Game Layers* permettent au concepteur de lister les différentes notions-clés de son futur jeu de manière hiérarchique. Ces couches sont réparties de la manière suivante :

- La couche *« TOKEN »*, tout en bas de la pile, est destinée à référencer tous les éléments de jeu, selon la *théorie des Tokens* que nous avons présentée précédemment. Là encore, nous observons que la notion de *« token »* est plutôt répandue dans l'industrie du jeu vidéo.
- La couche *« PROP »* permet de lister les différentes capacités, ou « propriétés », attribuées à chacun des *« tokens »* du jeu.
- La couche *« DYN »* recense les différentes actions proposées au joueur. Ces actions découlent directement des propriétés de la couche précédente, elles-mêmes associées à des éléments de jeu identifiés sur la première couche.
- La couche *« GOAL »* liste les objectifs que le joueur doit accomplir pour gagner au jeu.
- La couche *« META »* est destinée aux éléments a priori déconnectés des couches précédentes, mais qui jouent quand même un rôle dans l'interactivité ludique. L'auteur cite comme exemple la division d'un jeu en « niveaux » ou l'utilisation d'un nombre de vies limitées.
- La couche *« PSYCHO »*, dernière de la liste, vise tout simplement à recenser les différentes émotions ou sensations que le concepteur du jeu souhaite que le joueur ressente.

Même si sa philosophie globale est similaire, ce modèle n'est pas vraiment lié au *modèle MDA* : il utilise des couches véritablement différentes. Néanmoins, nous y retrouvons les notions de *« tokens »*, de *« Dynamics »* ou de « réponse émotionnelle du joueur » déjà aperçues dans d'autres outils présentés dans ce chapitre.



26. Représentation du jeu *Tetris* avec les *Game Layers*

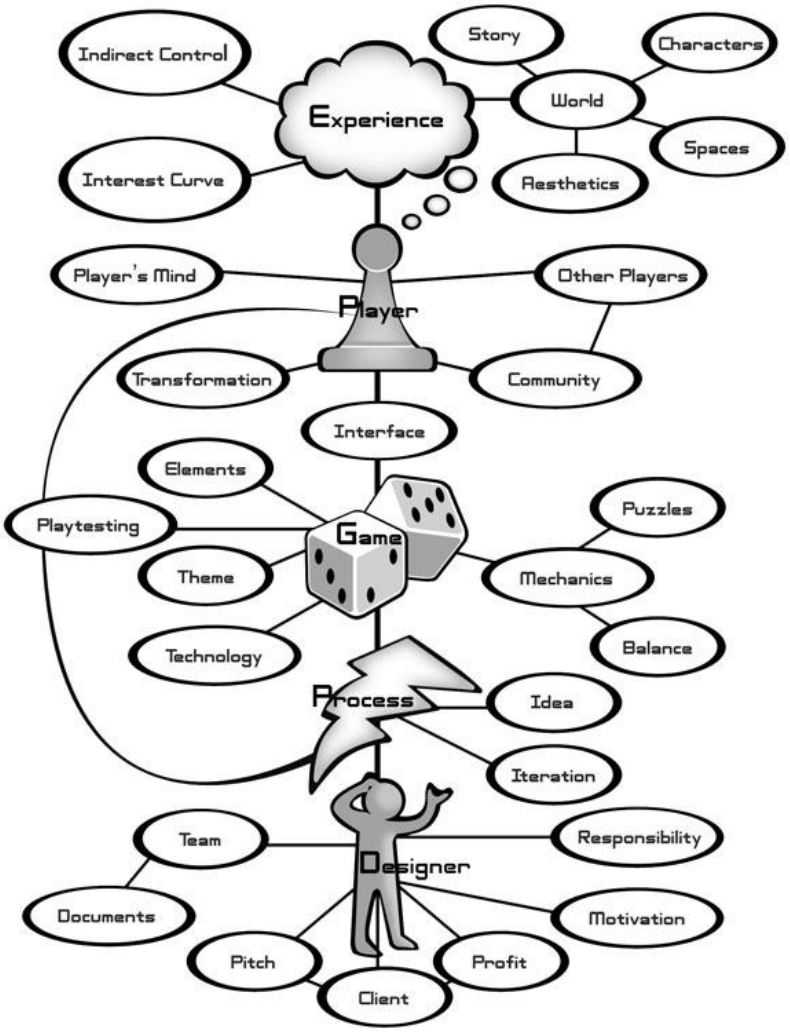
3.3.6 LES « LENSES » DE SCHELL

Si ces quatre derniers modèles proposent globalement d'analyser la relation « joueur-jeu » à travers des niveaux de lecture, le nombre de ces niveaux reste relativement restreint *(trois, six et neuf)*. L'idée est ici de pouvoir mettre les différents niveaux en relation tout en gardant un système simple à utiliser lors de la conception d'un jeu. ***Schell*** a reprit ce concept de niveau de lecture à travers ses *Game Design Lenses*, mais a décidé de sacrifier la relation directe entre chaque niveau de lecture pour proposer une quantité très élevée de *Lenses*. Son ouvrage (Schell, 2008b) décrit un total de cent *Lenses* focalisées sur des aspects très précis de la relation « joueur-jeu ». Bien qu'elles ne soient pas directement liées entre elles, elles sont rangées en cinq catégories. Ces dernières sont ensuite mises en relation par le biais d'une modélisation de la relation « joueur-jeu ». Plus précisément, ce modèle prend la forme d'un « plan » illustrant les points importants du processus de Game Design. Cinq éléments principaux, attachés à de nombreux sous-éléments, y sont présents :

- *« Designer »*, le concepteur du jeu, rattaché à des notions comme *« Motivation »*, *« Documentation »*, *« Profit »*, *« Equipe »*…
- *« Processus »*, le processus de création du jeu à proprement parler, connecté à seulement deux notions : *« Idée »* et *« Itération »*. Il est également relié à l'élément *« Joueur »* pour signifier son importance lors des tests.
- *« Jeu »*, l'objet jeu en lui-même, divisé en de nombreuses notions telles que *« Eléments »*, *« Interface »*, *« Equilibre »*, *« Technologie »*, *« Mécanismes »*…
- *« Joueur »*, relié au jeu à travers la notion *« Interface »*, possède également des sous-catégories propres telles que *« Communauté »*, *« Esprit »*, *« Transformation »*…
- *« Expérience »*, directement connecté au joueur, rassemble toutes les notions permettant de décrire l'expérience de jeu : *« Monde »*, *« Esthétique »*, *« Histoire »*, *« Courbe de Progression »*…

Dans le corpus étudié, le modèle proposé par ***Schell*** est le plus complet car il intègre aussi bien le processus de conception que l'expérience de jeu ressenti par le joueur, sans oublier les différents acteurs de l'interactivité ludique. Si ce modèle vise à théoriser la conception d'un

jeu en rappelant ses différents aspects de manière exhaustive, la notion de *Lenses* peut également être utilisée de manière bien plus appliquée pour le seul processus de conception, comme nous l'avons évoqué dans la section précédente *(p.61)*.



27. La représentation de la relation « joueur-jeu » par *Schell*

Au sein de notre corpus, 21 textes différents visent à proposer au moins un modèle formel à proprement parler. Les 19 textes restants s'articulent autour d'une volonté de théorisation moins poussée. Ils visent plutôt à prodiguer des « conseils de conception » génériques, voire à proposer tout simplement un retour d'expérience de la part de Game Designers expérimentés. Nous baptisons cette catégorie d'ouvrage des « livres de recettes ». Si la majorité d'entre eux sont de simples recueils de conseils, certains auteurs ont cependant cherché à formaliser leur façon de prodiguer ces conseils sous forme de « pièces à assembler ».

3.4 LES « PIECES A ASSEMBLER »

Cette catégorie regroupe six textes proposant des conseils sous forme de « pièces à assembler » afin de faciliter la conception d'un jeu vidéo.

3.4.1 LES « GAME DESIGN PATTERNS »

L'approche la plus structurée de cette catégorie consiste à proposer un ensemble de « stratégies de conception » basées sur une philosophie similaire aux *design patterns* utilisées en ingénierie informatique (Gamma, Helm, R. Johnson, & Vlissides, 1994). Les *Game Design Patterns* (Bjork & Holopainen, 2004) sont une collection de 200 « pièces » qui peuvent être utilisées pour concevoir un jeu. Si les deux chercheurs ont volontairement omis d'étayer leur collection de « patterns » par un modèle formel ou une définition du jeu, ils proposent

néanmoins une typologie hiérarchique destinée à classer les *« patterns »* selon les aspects du jeu qu'elles concernent : *« temporalité »*, *« règles »*, *« joueurs »*…

3.4.2 LE « 400 PROJECT » DE FALSTEIN

Dans une veine similaire, le *400 Project* lancé par ***Falstein*** (2006) vise à rassembler un grand nombre de conseils de conception formalisés comme des « règles à suivre ». Pour cela, ce Game Designer reconnu a appelé à contribution tous ses confrères. A ce jour, 112 règles sont proposées par ce projet. Si la philosophie du *400 Project* est similaire à celle des *Game Design Patterns*, les conseils proposés ne sont pas tout à fait les mêmes. Les *« patterns »* sont plutôt des « pièces de puzzle » à assembler pour construire un jeu, alors que les règles du *400 Project* sont des bonnes pratiques à suivre. En conséquence, ce dernier projet n'est appuyé par aucune « catégorisation » particulière des règles proposées.

3.4.3 D'AUTRES APPROCHES DE TYPES « PIECES A ASSEMBLER »

Si ces deux derniers textes proposent une véritable collection de « pièces à assembler » formalisées, l'idée même d'envisager une telle formalisation semble avoir été inspirée par la notion de *Formal Abstract Design Tool* introduite en 1999 par ***Church*** (2005). L'auteur y appelle à l'utilisation d'un langage uniforme et formalisé pour le Game Design, en s'appuyant sur deux ou trois exemples concrets. Cet appel a inspiré d'autres auteurs, tel que ***Leblanc*** (2005) qui propose des exemples « d'outils formels » destinés à créer de la tension dramatique dans les jeux.

Dans un autre registre, notons le travail de ***Koster***, qui propose d'utiliser des outils mathématiques pour la conception de jeux vidéo (Koster, 2009). Il s'appuie particulièrement sur les *problèmes NP-complets de Karp*, tels que *« l'empaquetage d'ensemble »* ou *« la couverture de sommets »*. Il montre qu'ils peuvent servir à résoudre des problématiques concrètes lors de la conception d'un système de jeu, par exemple lors de l'équilibrage des classes d'avatars dans un jeu de rôle massivement multijoueurs *(MMORPG)*.

Enfin, une dernière initiative de « formalisation » des conseils de Game Design se démarque du lot par sa référence au monde du cinéma. Inspiré par le *Dogme95* des réalisateurs danois[29], ***Adams*** (2001) propose le *Dogma 2001* pour le jeu vidéo. Il s'agit d'une série de dix « règles à suivre » pour créer des jeux vidéo. De l'aveu même de leur auteur, les règles proposées, telles que *« les scènes cinématiques non-interactives sont prohibées »* ou *« il pourra y avoir des gagnants, des perdants, des adversaires, mais pas de Bons absolus ou de Méchants absolus »* ne sont pas neutres. Elles reflètent un engagement artistique de la part d'***Adams***, qui appelle ainsi ses confrères à se focaliser sur l'expérimentation et le gameplay pour concevoir des jeux vidéo.

3.5 LES « CONSEILS DE CONCEPTION »

Enfin, au-delà de tous les textes déjà présentés, les treize éléments restants dans notre corpus proposent des « conseils de conception » nettement moins formalisés. Au regard de notre corpus, cette forme de textes aux propos purement empiriques est visiblement la plus répandue. Elle est représentée par les écrits de ***Saltzman*** (1999), ***Rouse*** (2001), ***Crawford*** (2003), ***Meigs*** (2003), ***Pedersen*** (2003), ***Rollings & Adams*** (2003), ***Bates*** (2004), ***Brathwaite & Schreiber*** (2008), ***Kremers*** (2009), ***Perry & DeMaria*** (2009), ***Albinet*** (2010), ***Trefry*** (2010) et ***Rogers*** (2010).

[29] Lancé par ***Lars von Trier*** et ***Thomas Vinterberg***, le *Dogme95* est un manifeste de dix règles à suivre pour réaliser un film. Ces règles reflètent un engagement artistique de la part des deux réalisateurs, qui furent suivis dans cette initiative par de nombreux autres réalisateurs à travers le monde.

Ces treize textes sont tous des ouvrages écrits par des Game Designers de l'industrie du jeu vidéo, à destination de leurs confrères. S'ils ne proposent pas d'outils théoriques d'aide à la réflexion, ils semblent néanmoins tous tenir d'une philosophie proche. Globalement, ces ouvrages sont organisés en chapitres consacrés à chacun des « point-clés » de la conception du jeu vidéo. Par exemple, un chapitre aborde la rédaction du *« Game Design Document »* *(p.44)*, un autre celui de la narration, du graphisme, du gameplay, de l'interface ou encore de l'intelligence artificielle. Ces ouvrages sont souvent accompagnés de comptes-rendus d'expériences de terrain, soit sous la forme d'interviews avec des Game Designers professionnels, soit par l'analyse de jeux existants. A noter que ces retours d'expérience sont également présents dans les textes des catégories précédentes lorsqu'il s'agit d'ouvrages.

Au final, en dépit de leur absence d'outils théoriques clairement formalisés, les « conseils de conception » restent riches d'enseignements sur la création d'un jeu vidéo.

3.6 APPLICATIONS DES OUTILS THEORIQUES D'AIDE A LA REFLEXION

A la lecture des textes de notre corpus, les outils théoriques d'aide à la réflexion que nous venons d'identifier semblent pouvoir être utilisés pour plusieurs finalités différentes.

3.6.1 GUIDE DE CONCEPTION

L'application la plus évidente de tous ces outils théoriques est bien évidemment celle d'aide à la conception de jeu vidéo. Tous ces outils sont inventés par des créateurs des jeux, et visent à faciliter le processus de Game Design. Ils emploient pour cela des approches très différentes. Certains proposent une typologie des éléments à construire, d'autres une méthode de représentation graphique, ou encore une catégorisation des profils de joueurs. De part leurs différences de fonctionnement et de résultats, ces outils s'avèrent complémentaires. Rien n'interdit à un concepteur d'en utiliser plusieurs à la fois.

Cependant, comme la complexité de certaines représentations schématique le laisse penser, ces outils théoriques sont loin d'être simples à utiliser. S'ils ont certes le potentiel de faciliter la création d'un jeu vidéo, en tout cas durant l'étape *« Imaginer »* du processus *(p.65)*, ce potentiel de facilitation semble difficilement accessible à des novices. Afin d'être en mesure d'utiliser les outils que nous avons présenté, il semble que le concepteur se doit déjà d'être familier avec un certain nombre de concepts théoriques liés à la création de jeu. Par exemple les notions « d'éléments de jeu », de « profils de joueurs », de « récompenses » ou encore de « règle » sont prises pour acquises dans la description des divers outils étudiés. L'approche de facilitation de la création vidéoludique représentée par ces outils théoriques d'aide à la réflexion semble donc être plutôt destinée à un public d'experts. Ces outils étant avant tout destinés aux professionnels, ce constat semble finalement assez logique. Les novices pourront toujours commencer par la lecture des ouvrages accompagnant ces modèles, dont les « conseils de conception » qu'ils prodiguent sont destinés à faciliter l'apprentissage du « Game Design », comme nous le verrons ci-après *(p.82)*.

Il convient également de rappeler que tous ces outils étant « théoriques », ils ne peuvent pas directement produire de jeux qui soient « jouables ». Quel que soit le support de jeu envisagé, tous ces outils peuvent être utilisés pour concevoir de manière théorique le fonctionnement du jeu. Mais pour pouvoir le tester avec des joueurs, il faudra ensuite passer par une étape de « fabrication ». Comme nous l'avons déjà évoqué *(p.44)*, dans le cas du jeu vidéo, cette étape de fabrication implique tout simplement de réaliser un programme informatique permettant de jouer au jeu.

3.6.2 GRILLE D'ANALYSE

Ces outils théoriques peuvent également contribuer à l'analyse de jeux, et non pas seulement à leur conception. Les modèles proposant des niveaux de lectures, tels que le *modèle MDA (p.75)*, les *Primary Schemas (p.75)*, les *Game Elements (p.76)*, les *Game Layers (p.77)*, les *Lenses (p.78)*, et ceux proposant une typologie *(p.68)* sont d'ailleurs explicitement présentés par leurs auteurs comme de potentielles grilles d'analyses de jeux existants. De plus, la plupart des outils présentés dans ce chapitre servant à construire des jeux, ils peuvent tout aussi bien être utilisés pour les déconstruire. Tous ces outils théoriques viennent donc enrichir les modèles d'analyse des chercheurs spécialisés en jeu vidéo (Rufat & Minassian, 2011).

3.6.3 VECTEUR PEDAGOGIQUE

La plupart des outils théoriques analysés ici sont originellement introduits par des articles de revues spécialisées ou dans des ouvrages. En plus de la présentation du fonctionnement de l'outil, ces ouvrages proposent des conseils de conception et autres réflexions sur certains aspects du Game Design, tels que le gameplay, l'équilibrage, l'interface… Si cela n'est pas forcément le cas des articles de revues à cause de leur taille réduite, nous retrouvons néanmoins dans leur style d'écriture une certaine volonté pédagogique. En d'autres termes, tous les textes étudiés dans notre corpus semblent avant tout pensés pour apprendre comment créer des jeux au lecteur. De part leur aspect formel, ces outils permettent de structurer une méthodologie de conception de jeu vidéo, et la rendent donc potentiellement plus accessible à de nouveaux apprenants.

De plus, le potentiel analytique de ces outils théoriques de conception constitue probablement une des réponses aux nombreux défis soulevés par l'enseignement de la culture ludique *(désignée en anglais par le terme « ludoliteracy »)*. Des études de terrain sur le sujet (Zagal, 2010; Zagal & Bruckman, 2008) font remonter qu'il est par exemple difficile d'inciter des étudiants à quitter leur posture de « joueur passionné » pour se mettre dans une posture propice à l'analyse de jeux. De même, les différences de niveaux de culture ludique au sein d'un groupe d'étudiants ont tendance à créer un public très hétérogène, ce qui complique la mission de l'enseignant. Nous pouvons alors imaginer que les outils théoriques s'appuyant sur un modèle formel pourraient apporter une réponse sur ces deux points. En proposant un cadre d'analyse clairement défini au travers d'un outil, les étudiants adopteront peut-être plus facilement une posture d'analyse. De plus, le fait que cet outil théorique soit inconnu de la plupart des étudiants permettra peut-être d'homogénéiser légèrement le niveau d'un groupe d'étudiants. Pris de manière isolée, ces outils théoriques d'aide à la réflexion semblent difficilement accessibles aux novices. Ils semblent donc plus adaptés à un contexte pédagogique où l'enseignant peut les accompagner dans leur apprentissage de l'outil.

3.6.4 OUTIL DE COMMUNICATION

Enfin, terminons par une proposition d'application nouvelle pour ces outils. Comme nous l'avons déjà évoqué, dans l'industrie vidéoludique le travail d'un Game Designer consiste à produire un cahier des charges appelé *« Game Design Document » (p.46)*. Ce document exhaustif détaille de manière très précise le fonctionnement du jeu inventé par le concepteur, afin de permettre aux autres membres de l'équipe *(programmeurs, artistes…)* de le fabriquer. Les gros studios de création de jeu vidéo sont d'ailleurs organisés en « départements » spécialisés dans chacun des domaines impliqués : conception, programmation, graphisme, son, tests… En tant que premier élément de la chaîne de fabrication d'un jeu vidéo, le département de conception est régulièrement amené à communiquer le fruit de son travail aux autres départements. Ce rôle est traditionnellement tenu par le *« Game Design Document »* (Rogers, 2010; Rouse, 2001). Cependant, une récente étude menée au sein des studios de création vidéoludique montre une évolution de cette pratique (Hagen, 2010). En effet, les studios constatent aujourd'hui que le *« Game Design Document »* est de moins en moins utile car il est trop volumineux, ce qui décourage les employés de le lire. Ce constat reflète

l'évolution des pratiques de production de jeu vidéo, telles que l'abandon du modèle de développement en cascade au profit de méthodes agiles *(p.94)*. Les studios cherchent donc aujourd'hui des moyens de communication plus efficaces entre leurs différents départements. D'après les studios interrogés par l'étude, aucun standard n'est actuellement établi en la matière. La tendance est à l'expérimentation de nouveaux moyens de communication adaptés aux échanges entre chaque type de département.

Dans ce climat d'expérimentation, nous pensons que certains outils de notre corpus pourraient constituer des moyens de communication fort intéressants. En particulier, nous pensons que les modèles formalisant le jeu sous forme de diagrammes peuvent faciliter la communication entre le département de conception et celui de programmation. En effet, certains de ces modèles formalisent le jeu selon des standards appartenant à la culture de l'ingénierie logicielle, telle que le langage *UML* pour *Machinations* *(p.70)* ou les *réseaux de Pétri* pour la *Game Grammar* *(p.69)*. Mais même quand ils ne relèvent pas d'un standard, ces modèles formels semblent tous fortement influencés par le paradigme de programmation *« Orienté Objet »*. Par exemple, les *Tokens* *(p.68)* proposent de diviser un jeu en un ensemble d'éléments sous forme d'une hiérarchie qui n'est pas sans rappeler le principe d'héritage du *« modèle Orienté Objet »*. Certains outils pourraient également s'avérer intéressants pour la communication entre les départements de conception et de tests utilisateurs. Par exemple, la *Game Alchemy* *(p.73)* et les *Game Layers* *(p.77)* permettent tous deux de formaliser les émotions que le concepteur souhaite faire ressentir au joueur. Ces deux outils théoriques d'aide à la réflexion pourraient alors servir à produire un document que le département de tests utiliserait comme référence lors de ses évaluations.

S'il ne s'agit là que d'une perspective théorique qui reste encore à valider sur le terrain, nous pensons vraiment que ces quelques outils théoriques de Game Design ont un réel potentiel pour la communication interne des studios de développement de jeu vidéo.

4 SYNTHESE : OUTILS THEORIQUES ET FACILITATION DU GAME DESIGN

Dans ce chapitre, nous nous sommes intéressés aux outils théoriques visant à faciliter le Game Design. Pour cela, nous avons constitué un corpus de texte relatifs au *« Game Design »*, et les avons étudié à la recherche d'outils théoriques *(p.59)*. Nous en avons identifié deux sortes : les méthodologies de conception *(p.60)* et les outils d'aide à la réflexion créative *(p.68)*.

En ce qui concerne les méthodologies de conception, nous observons plusieurs modèles du processus de création d'un jeu vidéo. Bien que chacun de ses modèles soit basé sur une « série d'étapes » différente, nous avons pu identifier des récurrences dans les douze modèles que nous avons étudiés. Ces récurrences nous ont permis de proposer le *modèle générique ICE (p.65)*, un cadre permettant de définir des « série d'étapes » du processus de création vidéoludique. Reposant sur un cycle itératif, ce modèle défini trois « étapes génériques » : *« Imaginer »*, *« Créer »* et *« Evaluer »*. Les méthodologies de conception que nous avons étudié proposent alors une ou plusieurs étapes qu'il est possible d'inscrire dans ces « étapes génériques ». Le *modèle générique ICE* peut donc être vu comme un cadre synthétique à partir duquel chaque créateur peut définir sa propre « série d'étapes » pour créer un jeu vidéo.

Nous avons également identifié des outils d'aide à la réflexion créative, destinés à guider la démarche du créateur durant l'étape générique *« Imaginer »*. A cette fin, ces outils proposent plusieurs approches différentes. Certains outils proposent des méthodes de modélisation graphique de la structure d'un jeu *(p.68)*. D'autres proposent d'analyser les joueurs *(p.71)* et la manière dont ils interagissent avec un jeu *(p.73)*. Enfin, une dernière approche vise à proposer des « pièces à assembler » permettant de créer un jeu plus rapidement *(p.79)*. S'ils sont principalement destinés à faciliter la création de nouveaux jeux, nous observons également que ces outils théoriques semble pouvoir servir à d'autres applications, telle que l'analyse de jeux existants ou l'apprentissage de la création vidéoludique *(p.81)*. Mais quelle que soit l'application envisagée, ces outils d'aide à la réflexion créative semble relativement complexes à utiliser. S'ils représentent une approche de facilitation de la création de jeu vidéo, cette facilitation se destine principalement à des créateurs « experts ». Au vu de l'origine professionnelle de ces outils, ce constat était malheureusement prévisible. Mais les amateurs ne possédant pas d'outils théoriques qui leur sont destinés *(p.57)*, seule l'exploration des outils professionnels pouvait nous permettre d'identifier des approches théorique de facilitation du Game Design. Nous essaierons néanmoins d'éprouver le potentiel de ces outils théorique d'aide à la conception pour la facilitation du « Serious Game Design », autant dans un contexte industriel *(p.87)* que pédagogique *(p.248)*.

Que ce soit pour les méthodologies de conception comme pour les outils d'aide à la réflexion créative, nous constatons également que ces outils sont aussi nombreux que différents. Cette multitude d'outils semble très surprenante au premier abord, d'autant plus que plusieurs outils s'appuient sur un modèle formel. Il semble que cet effort de formalisation des outils théorique ne leur permette pas pour autant de prétendre à l'universalité. Certes, nous observons des outils à la finalité différente. Par exemple, ceux visant à proposer un modèle formel du jeu s'avèrent complémentaires de ceux classifiant les joueurs. Mais même au sein d'une même catégorie d'outils, plusieurs modèles existent. Par exemple, nous recensons sept modèles formels différents pour la structure ludique. Comment pouvons-nous expliquer une telle multitude d'outils pour un même aspect de la création vidéoludique ? Les propos de ***Crawford*** (1982), lorsqu'il présente sa propre méthodologie de création de jeux vidéo, semblent nous apporter une réponse :

> *« La méthodologie que je décris est basée sur mes propres expériences de conception de jeu, et reflète la plupart des pratiques que j'utilise pour créer un jeu. Cependant, je n'ai*

> *jamais utilisé cette méthodologie pas-à-pas, et je ne recommande à personne de le faire. En premier lieu, le Game Design est une activité bien trop complexe pour être réduite à une simple procédure formelle. De plus, la personnalité du concepteur doit dicter sa manière de travailler. Mais surtout, le concept même de formalisation absolue s'oppose à la créativité qui caractérise le Game Design. […] Ainsi, je propose ma méthodologie non pas comme une formule normative, mais comme un ensemble d'habitudes et de pratiques que l'apprenti Game Designer pourra essayer d'intégrer à sa propre méthodologie de travail. »*[30] *[p.45]*

Ainsi, le plus ancien texte de notre corpus apporte déjà une réflexion très pertinente sur la potentielle formalisation du Game Design. S'il est possible de proposer des outils formels accompagnant le processus du Game Design, voire de formaliser le processus lui-même à travers une série d'étapes clairement définie, aucune de ces tentatives de formalisation ne peut avoir vocation à l'universalité. Comme le rappelle ***Crawford***, la nature créative du processus de conception d'un jeu implique une certaine part de liberté. Cela explique donc pourquoi il est possible d'identifier autant d'outils théoriques différents pour le Game Design : chacun reflète une part de la personnalité de son inventeur. La diversité des outils étudiés dans ce chapitre permet donc de répondre aux besoins d'un large panel de Game Designers. Et si aucun des outils recensés ici n'est satisfaisant pour un concepteur donné, il lui sera toujours possible de s'approprier un modèle existant en le transformant, voire d'en créer un complètement original.

Au final, les propos de ***Crawford*** mettent en lumière le principal intérêt de recenser les outils théoriques de Game Design. Comme il n'existe pas d'outil théorique universel, il est essentiel de recenser un large panel d'approches existantes afin d'accompagner de nombreux concepteurs différents dans leur démarche créative. Nous supposons donc qu'il existe également plusieurs outils théoriques adaptés aux Serious Games. Mais, dans cette hypothèse, ces outils théoriques destinés au Serious Game Design sont-ils différents de ceux que nous venons de recenser pour le Game Design ? Si oui, est-il alors possible d'utiliser ces outils théoriques de Game Design pour la création de Serious Games ?

[30] *"The procedure I will describe is based on my own experiences with game design, and reflects many of the practices that I use in designing a game. However, I have never used this procedure in a step-by-step fashion, nor do I recommend that any person follow this procedure exactly. In the first place, game design is far too complex an activity to be reducible to a formal procedure. Furthermore, the game designer's personality should dictate the working habits she uses. Even more important, the whole concept of formal reliance on procedures is inimical to the creative imperative of game design. […] I therefore present this procedure not as a normative formula but as a set of suggested habits that the prospective game designer might wish to assimilate into her existing work pattern."*

METHODOLOGIES DE CONCEPTION DE SERIOUS GAMES

Par analogie à la définition du *« Game Design »* *(p.42)*, nous avons défini le *« Serious Game Design »*, comme le processus permettant de créer un Serious Game *(p.56)*. En plus de son « résultat », un processus se défini aussi par une « série d'étapes » *(p.60)*. S'il n'existe pas de « série d'étapes » universelle pour la conception d'un jeu vidéo *(p.65)*, en existe-t-il une pour la conception de Serious Games ?

Pour tenter de répondre à cette question, nous proposons d'étudier les « outils théoriques » pensés pour la conception de Serious Games. D'une manière analogue à la sphère du jeu vidéo de divertissement *(p.59)*, la majorité de ces outils théoriques de Serious Game Design sont issus ou destinés aux professionnel de l'industrie.

1 RETOUR D'EXPERIENCE : LE PROCESSUS DE CONCEPTION DE GEDRIVER

Nous proposons de commencer cette étude des outils théoriques de Serious Game Design par un retour d'expérience. Dans le cadre du *CIFRE*, nous avons été amené à travailler au sein de la société ***OKTAL***. Cette entreprise est spécialisée dans la réalisation de Serious Games et autres simulations visant à prodiguer un entraînement, pour quatre domaines d'application : l'automobile, l'aviation, le ferroviaire et la défense. Possédant de nombreuses années d'expérience dans les simulateurs de formation professionnelle, ***OKTAL*** était maître d'œuvre d'un projet de Serious Game visant à sensibiliser les conducteurs à l'éco-conduite. Baptisé *geDriver*, ce projet est doté d'un budget total de 1.2 millions d'euros. Une partie de ce budget provient d'un financement public, ce projet ayant été sélectionné par le « volet numérique » du plan de relance économique initié par ***Nathalie Kosciusko-Morizet*** en mai 2009. Le projet *geDriver* rassemble un consortium de cinq partenaires : le constructeur automobile ***Renault***, la société ***Key Driving Competences (KDC)*** spécialisée dans la réalisation d'accessoires de mesure pour l'analyse du comportement de véhicules, les laboratoires ***LEAD*** et ***ARTS*** de l'***université de Bourgogne***, et ***OKTAL***. En associant des interfaces réalistes *(volant, pédales, levier de vitesse d'un véritable véhicule…)* et l'outil technique de création de simulations automobiles *SCANeR* *(p.215)*, l'objectif est de réaliser un simulateur permettant au grand public et aux professionnels d'apprendre les gestes de l'éco-conduite. Ces gestes sont destinés à réduire l'émission de *CO2* d'un véhicule en modifiant les habitudes de son conducteur. Une des problématiques de recherche explorées par ce projet est l'évaluation de la valeur ajoutée de l'approche « Serious Game » pour capter l'intérêt des apprenants. Afin que les laboratoires de recherche puissent étudier le comportement des utilisateurs, la société ***OKTAL*** a reçu pour mission de concevoir et réaliser un prototype de Serious Game prodiguant un entraînement aux gestes de l'éco-conduite.

Nous avons été amenés à participer directement au processus de conception de *geDriver*, en collaboration avec les ingénieurs de la société ***OKTAL***. Les différentes étapes de ce processus sont détaillées ci-après.

1.1 DEFINITION DU CONTENU « SERIEUX ».

Le contenu « sérieux » du jeu a tout d'abord été formalisé lors de réunions et échanges avec les autres partenaires. A partir de documents de références sur le sujet, tels que les guides proposés par l'***ADEME***, une liste de gestes d'éco-conduite à transmettre à travers le jeu a été définie. Cette liste s'est accompagnée de la sélection des véhicules proposés dans le jeu, en étroite collaboration avec ***Renault*** qui fournissait les données permettant de simuler le comportement desdits véhicules : consommation, courbe de rejet de *CO2* selon le régime moteur… Lors de cette phase de réflexion préliminaire, il a également été décidé d'ajouter un autre contenu « sérieux » au jeu : une sensibilisation à la sécurité routière.

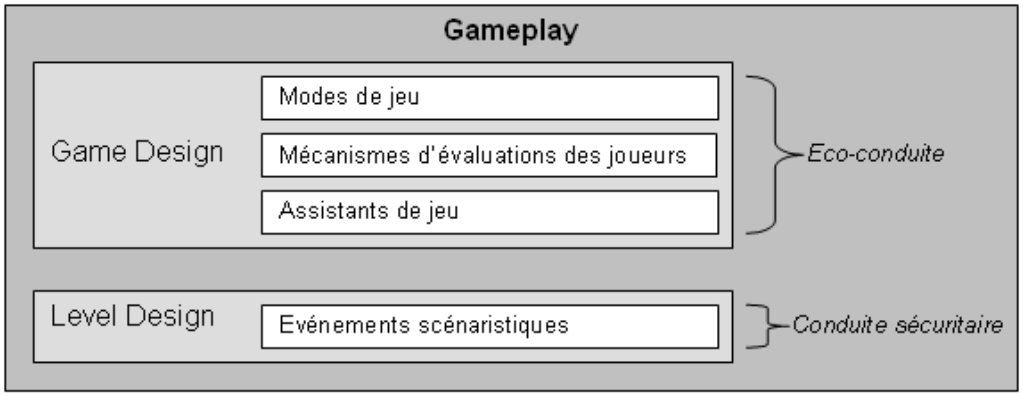
Au final, les partenaires ont défini une liste exhaustive du « contenu sérieux » auquel ce Serious Game permet de s'entraîner : les gestes permettant de limiter le rejet de *CO2* du véhicule *(extinction du moteur lors d'un arrêt prolongé à un feu rouge, bonne utilisation des rapports de vitesse…)* et les habitudes de conduite permettant d'éviter les accidents *(anticipation du freinage, contrôle des rétroviseurs, respects des limitations de vitesse…)*.

1.2 GAME DESIGN : DEFINITION D'UN CONCEPT DE JEU

A partir du contenu « sérieux » défini par les partenaires, la seconde étape consiste à inventer un concept de jeu permettant de transmettre ce contenu sérieux. Ce concept de jeu et la description détaillée de ses mécanismes seront ensuite formalisés sur « papier », afin que les développeurs puissent fabriquer des prototypes *(p.44)*.

OKTAL ne possédant pas d'outils théoriques d'aide à la réflexion créative particulier, nous avons proposés l'utilisation de ceux issus du Game Design pour cette phase *(p.68)*. La conception des mécanismes de jeux étant principalement assurée par une seule personne, l'usage pressenti pour ces modèles formels était la communication du concept aux autres membres de l'équipe travaillant sur *geDriver*. Cependant, les outils que nous estimions comme pertinents pour cette tâche *(p.82)*, se sont révélés trop complexes pour les besoins de communication du projet. En effet, ce projet réunit de nombreux partenaires qui ne possèdent pas de compétence particulière en matière de conception vidéoludique mais qui doivent néanmoins être en mesure de valider régulièrement le fonctionnement du jeu. Nous avons alors dû nous tourner vers une approche plus simple. En nous inspirant des outils théoriques existants, nous avons essayé de « vulgariser » certains concepts relatifs à la conception de jeu vidéo, tout en exposant les différents principes de jeu proposés, ainsi que leurs liens avec le contenu sérieux. Le cahier des charges du projet introduit la description des mécanismes de jeu de la manière suivante :

> *La proposition consiste en la spécification de plusieurs « Gameplay », autrement dit de plusieurs « jeux » différents. Ces gameplay sont composés de deux composantes principales : le « Game Design » et le « Level Design ». Chacune de ces composantes est constituée de plusieurs parties qui seront spécifiées distinctement dans ce document : les modes de jeu, les mécanismes d'évaluations, les assistants de jeu et les évènements scénaristiques.*
>
> *Le traitement de l'aspect éco-conduite repose principalement sur le Game Design, et celui de la conduite sécuritaire sur le Level Design.*



Selon ce principe, trois propositions de gameplay sont spécifiées dans le présent document :

1) Le gameplay « Taxi », basé sur un trajet à faire en temps limité.

2) Le gameplay « Marathon » (option), basé sur une course à la distance avec essence limitée.

3) Le gameplay « Dragster » (option), centré sur le bon passage des rapports de vitesse.

Ensuite, pour chaque *« Gameplay »*, les quatre composantes *(modes de jeu, mécanismes d'évaluations, assistants de jeu et évènements scénaristiques)* sont détaillées. Pour les modes de jeu, un rappel des règles communes est d'abord fait de manière textuelle. Par exemple, pour le gameplay *« Taxi »*, les règles sont les suivantes :

- Le joueur est placé aux commandes d'un véhicule léger et a pour mission d'atteindre une destination précise dans un temps limité. La mission du joueur est d'amener son bébé à la crèche.

- Si le joueur n'atteint pas la destination dans le temps imparti, il a perdu.

- S'il l'atteint, on lui propose alors un score, calculé à partir du CO2 qu'il a émis. En effet, durant le trajet, la production de CO2 est évaluée en temps réel. Elle détermine donc directement le score du joueur : plus le taux de CO2 émis durant son trajet sera faible, meilleur sera son score.

- L'enfant sera visible dans le rétroviseur du véhicule : il sera en train de dormir sur un siège bébé à l'arrière du véhicule. La « dangerosité » de la conduite du joueur influera directement sur le comportement du bébé. Tant que le joueur roule « prudemment », le bébé dormira. Si le joueur a des réactions brusques, par exemple face à des événements scénaristiques, cela réveillera le bébé. Si ce dernier a encore peur après s'être réveillé, il se mettra à pleurer. Si le joueur atteint l'objectif dans le temps imparti, un bonus/malus de score sera appliqué selon le comportement du bébé : -100 points s'il pleure, 0 points s'il est juste réveillé, et +100 points s'il dort encore.

A ces règles générales s'ajoutent des règles spécifiques aux différents modes de jeu proposés. Pour le gameplay *« Taxi »*, ils sont au nombre de deux : un mode *« mission »*, basé sur le franchissement de niveaux, et un mode *« score »*, dans lequel le joueur doit réaliser le meilleur score possible sur un niveau unique. En plus de la description textuelle de chacune des règles spécifiques au mode de jeu, un tableau de synthèse est proposé afin de faciliter la compréhension du Game Design par tous les partenaires qui doivent l'approuver. Ce tableau

s'appuie sur un formalisme volontairement très simple, dérivé de nos travaux antérieurs sur la définition du Gameplay au niveau formel (Djaouti, J. Alvarez, Jessel, & Methel, 2008) :

> *« Le gameplay naît de l'association de « règles Game », qui définissent des objectifs à atteindre, et de « règles Play », qui définissent des moyens et contraintes pour atteindre ces objectifs. »*[31]

Pour chaque mode de jeu, nous présentons ainsi de manière séparée les **objectifs** permettant gagner le jeu, les **moyens** proposés au joueur pour atteindre ces objectifs, et les **contraintes** qui visent à l'en empêcher :

Mode de jeu	Objectifs	Moyens	Contraintes
Mission *(apprentissage)*	*Atteindre un point donné *Réaliser un score minimal pour passer au niveau suivant	*Conduire un véhicule *Appliquer les gestes d'éco-conduite	*Temps limité *Rejeter le moins de CO2 possible

28. Exemple d'un tableau de synthèse utilisé dans le cahier des charges de *geDriver*

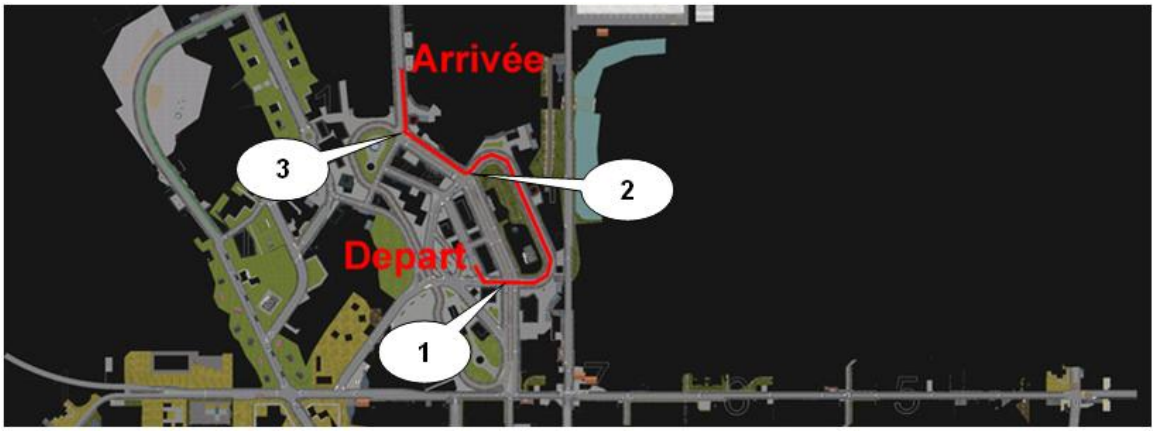
Le cahier des charges définit ensuite les modalités d'évaluation du joueur *(algorithmes de calcul du score…)* ainsi que les différents types d'évènements scénaristiques pouvant survenir durant les niveaux de jeu. Ces évènements scénaristiques sont utilisés pour l'étape suivante de la conception du Serious Game.

1.3 LEVEL DESIGN : CONCEPTION DES DIFFERENTS NIVEAUX DU JEU

Une fois les mécanismes de jeu approuvés par tous les partenaires, vient la seconde phase de conception : le *« Level Design »*. Elle consiste à inventer des scénarios mettant en scène les mécanismes de jeu. Concrètement, il s'agit de construire les différents « niveaux » du jeu, d'où le nom donné à cette phase *(p.45)*.

Pour cette phase du projet, ***OKTAL*** s'est appuyée sur les méthodes de conception qu'elle a l'habitude d'utiliser pour la réalisation de ses projets de simulateurs pédagogiques. La conception des niveaux a eu lieu sur « papier », dans un document *Word* respectant un formalisme bien défini. Tout d'abord, un plan du réseau routier où se déroule le niveau est inséré dans le document. Sur cette image, le concepteur définit le parcours que doit suivre le joueur en indiquant un point de départ et d'arrivée et en surlignant le ou les chemins à parcourir. Ce parcours est ensuite annoté avec des numéros correspondants aux différents évènements pouvant survenir lors du scénario. Sous l'image, se trouve un tableau détaillant chacun des évènements : *description textuelle, liste des pictogrammes à afficher à l'écran, sons à jouer pour accompagner l'évènement…* Pour ce projet nous avons rajouté une légère subtilité au formalisme inventé par ***OKTAL***. Les descriptions des évènements scénaristiques liés à l'éco-conduite sont rédigées en vert, alors que ceux relatifs à la sécurité sont écrits en rouge.

[31] *« Gameplay emerges from the association of "Game rules", stating a goal to reach, with "Play rules", defining means and constraints to reach this goal. »*



Indice	Situation	Audio		Info visuelle		Evènements (Ecologie – Sécurité)
		Son	Mode	Picto	Mode	
Départ						Le joueur doit démarrer la voiture
1						Feu qui passe au rouge quand le joueur arrive : incitation à couper son moteur
2						Feu qui est déjà rouge quand le joueur arrive, mais feu « long » : incitation à couper son moteur au bout de quelques temps où l'on voit que le feu ne passe pas vert

29. Exemple du formalisme utilisé pour la conception des niveaux de *geDriver*

Suivant ce formalisme nous avons inventé différents « niveaux » pour *geDriver*. Certains de ces niveaux comportent des « variantes » selon le véhicule utilisé, comme des situations spécifiques à la conduite d'un semi-remorque.

1.4 REALISATION ET EVALUATION DE PROTOTYPES

Une fois les étapes de conception « papier » terminées, la suite du processus implique de réaliser un prototype pour évaluer la pertinence du Serious Game. Ce processus est itératif. Un premier prototype est réalisé puis testé. A partir du retour des utilisateurs, le concepteur modifie son design. Un autre prototype est donc réalisé à partir du design corrigé, puis réévalué. Ce processus se poursuit jusqu'à ce que le retour des utilisateurs corresponde aux objectifs visés par le concepteur. Dans le cas de *geDriver*, il s'agit d'évaluer si le contenu sérieux défini au début de la réflexion est effectivement transmis par le jeu. Cette étape a été confiée aux laboratoires de recherche impliqués dans le projet. Le rôle d'***OKTAL*** s'est ici limité à fournir un outil technique permettant la réalisation des prototypes, ainsi qu'à en réaliser une première ébauche pour aider les chercheurs. Nous n'avons donc pas été directement impliqué dans cette étape. Nous avons par contre eu l'opportunité de travailler plus particulièrement sur *SCANeR*, l'outil technique de création de Serious Games développé par ***OKTAL***, au cours d'un autre projet qui sera détaillé dans la troisième partie *(p.215)*.

1.5 SYNTHESE DE CE RETOUR D'EXPERIENCE

Au final, le processus utilisé lors de la réalisation de ce projet de Serious Game, par un acteur de son industrie, fait ressortir de nombreux points commun avec les modèles du processus de Game Design étudiés précédemment *(p.60)*. Par exemple, nous retrouvons la notion de « processus itératif » par la réalisation et l'évaluation successive de prototypes. Nous retrouvons également une phase de *« Game Design »* dédiée à l'élaboration des mécanismes de jeu, couplée à une phase de *« Level Design »* permettant d'inventer les différents

« niveaux » ou « scénarios » qui mettent en scène ces mécanismes de jeux[32]. Si nous avions précédemment identifié de nombreux outils théoriques d'aide à la réflexion destinés à faciliter la conception de la phase *« Game Design »* *(p.68)*, nous n'avions par contre pas identifié d'outils véritablement dédiés au *« Level Design »*. Même les ouvrages consacrés uniquement à cet aspect de la conception vidéoludique (Kremers, 2009) rentrent pour l'instant dans notre catégorie « conseils de conception » *(p.80)*. Le processus de conception de *geDriver* comble cette lacune en faisant appel aux méthodologies éprouvées par ***OKTAL*** lors de la conception de simulateurs pédagogiques. Le *« Level Design »* de ce projet est donc réalisé selon un formalisme interne à la société. Ce formalisme complète le modèle théorique utilisé pour la partie *« Game Design »*, qui lui est inspiré des outils théoriques destinés au jeu vidéo de divertissement. Ce retour d'expérience met donc en évidence un processus de « Serious Game Design » qui se rapproche de ceux du « Game Design » par certains points, et s'en démarque par d'autres. Cette observation est-elle valable pour tous les modèles théoriques de création d'un Serious Game ?

2 D'AUTRES METHODOLOGIES DE CONCEPTION DE SERIOUS GAMES

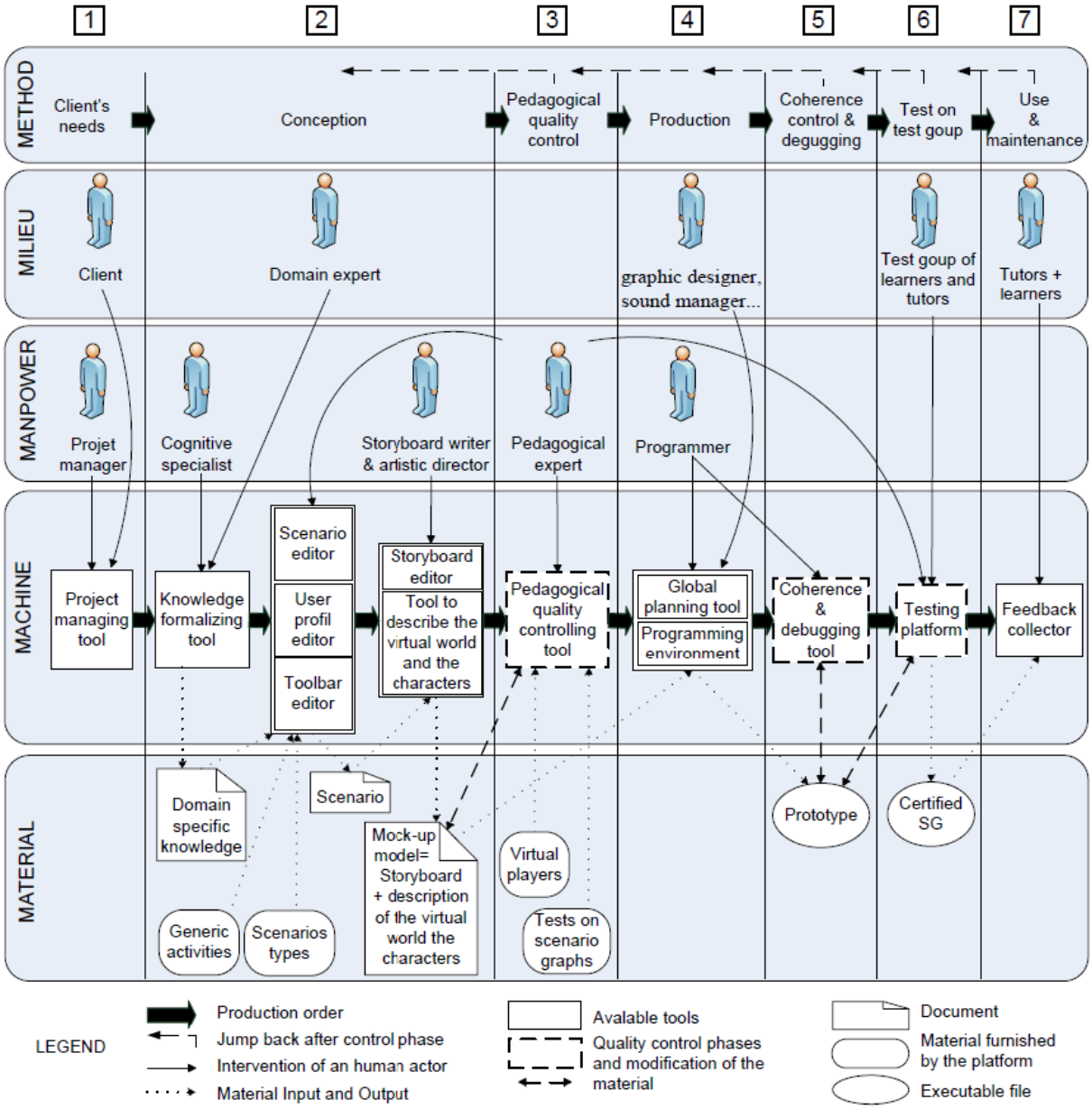
A l'image de ce que nous avons observé pour le jeu vidéo de divertissement, il existe plusieurs modèles du processus de conception de Serious Games. Nous avons pu identifier plusieurs outils théoriques manifestement destinés au *« Serious Game Design »*. Contrairement à ce qui a été observé pour le champ du *« Game Design »*, la plupart des outils identifiés rentrent dans la catégorie « méthodologie de conception » *(p.60)*. Seuls deux des onze outils que nous avons recensé sont des « outils d'aide à la réflexion créative » *(p.68)*. Un Serious Game ne se différenciant pas d'un jeu vidéo au niveau formel *(p.56)*, nous pouvons alors supposer que les outils théoriques de Game Design développés pour formaliser « l'objet-jeu » *(p.68)*, le « joueur » *(p.71)*, ou la « relation joueur-jeu » *(p.73)* restent applicables au Serious Game Design. Cela est d'ailleurs illustré dans une certaine mesure par notre retour d'expérience sur le projet *geDriver*. En revanche, nous pouvons supposer que le processus de conception en lui-même se distingue de celui d'un jeu vidéo de divertissement, ce qui expliquerait le nombre de travaux relatifs à cette question.

A ce jour, nous avons pu identifier onze outils théoriques dédiés au Serious Game. Ces travaux sont principalement issus de publications académiques *(ouvrages, mémoires, articles…)*, bien qu'ils soient parfois le fruit de collaboration entre entreprises et universités.

2.1 UN MODELE INDUSTRIEL REPOSANT SUR L'INGENIERIE PEDAGOGIQUE

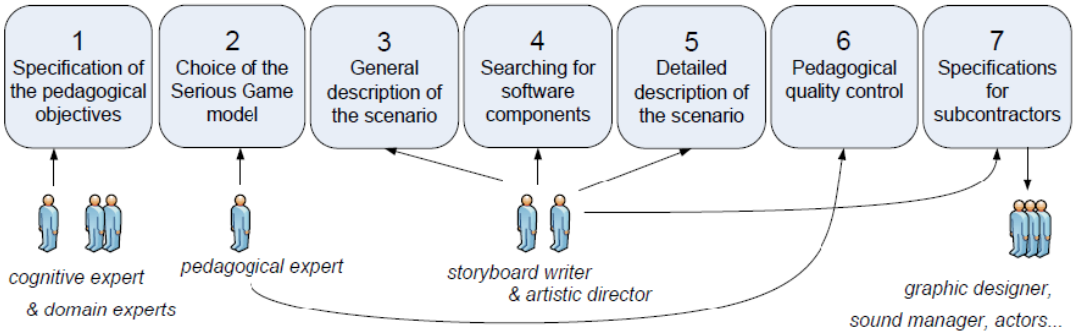
L'équipe de chercheurs travaillant sur *Learning Game Factory* *(p.39)* s'est appuyée sur ce projet pour développer des outils théoriques et techniques destinés à la conception de Serious Games. Si les outils techniques ne sont pas encore disponibles à la date d'écriture de cette thèse, les outils théoriques ont été présentés à travers des articles de recherche. Concrètement, ces chercheurs proposent deux modèles liés au processus de Serious Game Design. Tout d'abord, un premier modèle détaille le circuit industriel complet de création d'un Serious Game (Marfisi-Schottman, Sghaier, George, Tarpin-Bernard, & Prévôt, 2009) :

[32] Rappelons que si le terme *« Game Design »* désigne le processus global de conception d'un jeu, les professionnels du jeu vidéo opèrent généralement une distinction entre ses deux étapes principales *(p.45)*. Le terme *« Game Design »* peut donc également renvoyer à une partie seulement du processus de conception de jeu.



30. Modèle industriel de création d'un Serious Game par *Marfisi-Schottman & al.*

Ce modèle est associé à un autre travail similaire, mais qui se focalise uniquement sur les étapes de conception d'un Serious Game, en particulier pour les jeux destinés à transmettre un contenu pédagogique (Marfisi-Schottman, George, & Tarpin-Bernard, 2010). Nous pouvons voir les sept étapes du modèle ci-dessous comme une vue détaillée des trois premières étapes du modèle précédent :



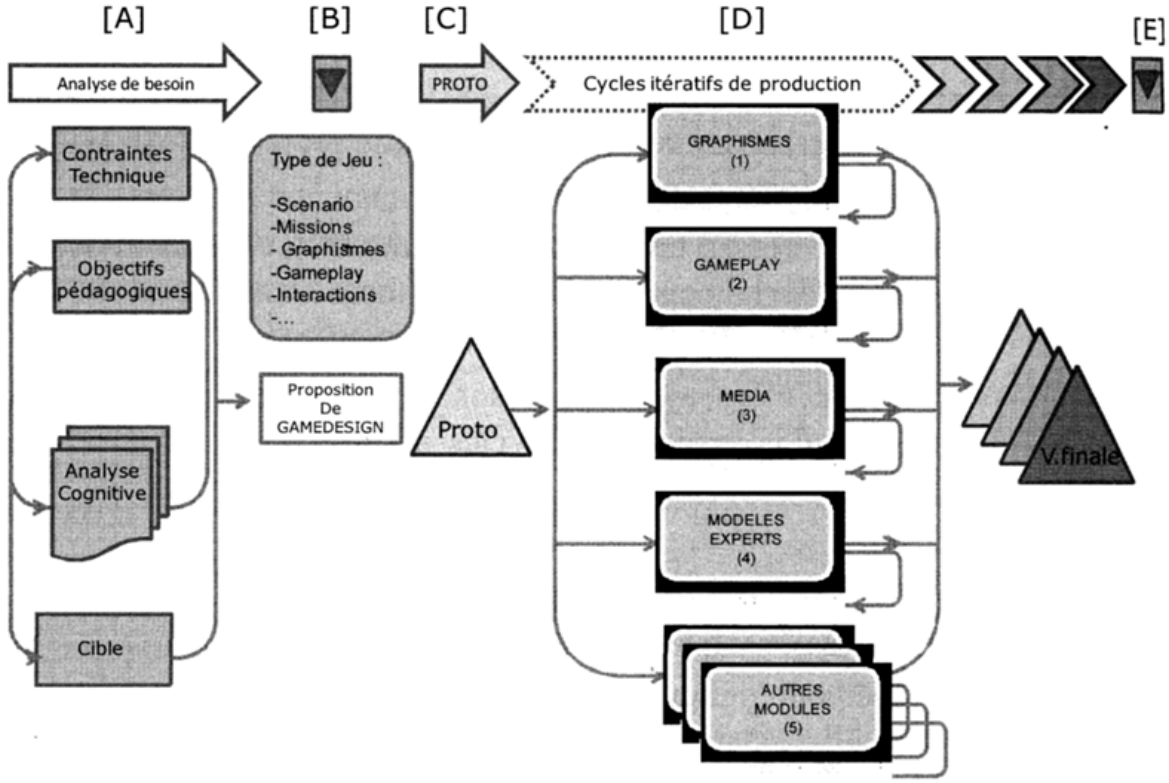
31. Modèle de conception d'un Serious Game par *Marfisi-Schottman & al.*

Globalement, ce modèle part de la spécification d'un contenu sérieux avec un « client », puis fait appel à des pédagogues pour concevoir un Serious Game dont le fonctionnement sera détaillé à travers de nombreux documents. Les chercheurs précisent avoir développé un outil technique destiné à supporter les outils théoriques qu'ils ont inventés. Il s'agit visiblement d'un tableau pouvant accueillir plusieurs *« widgets »*[33], chacun de ces *widgets* étant destiné à formaliser un des aspects du Serious Game. Un *widget* permet par exemple de lister toutes les « compétences » que doit transmettre le jeu, formalisant ainsi le « contenu sérieux » du jeu. D'une manière similaire, un autre *widget* permet de créer les scénarios de jeux selon le formalisme *IMS Learning Design*. Ce modèle formel d'ingénierie pédagogique permet de représenter des structures pédagogiques. Il est divisé en trois composantes principales : les acteurs pédagogiques se voient confier des *« rôles »* afin de permettre le déroulement *« d'activités »* dans des *« environnements »* spécifiques. Le principal avantage de formaliser ainsi un scénario pédagogique est la possibilité de les évaluer automatiquement grâce à des outils techniques dédiés. Le modèle de conception proposé par ***Marfisi-Schottman & al.***, s'appuie d'ailleurs ouvertement sur cet aspect lors de l'étape *« contrôle de la qualité pédagogique »*.

Comme nous pouvons le déduire de l'absence d'un « Game Designer » dans les acteurs impliqués pour la création de Serious Games selon ce modèle, le processus formalisé ici est inspiré par l'ingénierie pédagogique et se focalise principalement sur la création de jeux vidéo pour le domaine de l'éducation. Nous y retrouvons néanmoins des notions communes aux autres modèles, telles que le recours à un processus itératif ou la définition d'un contenu sérieux en préalable à la conception du jeu lui-même.

2.2 LE MODELE INDUSTRIEL DE KTM ADVANCE

Forte de son expérience dans le domaine, la société ***KTM Advance*** dispose également d'un modèle de conception industriel propre aux Serious Games (Boudier & Dambach, 2010).



32. Modèle industriel de création d'un Serious Game utilisé par *KTM Advance*

[33] Contraction de « window » et « gadget », un « widget » est un petit outil logiciel destiné à une tâche simple.

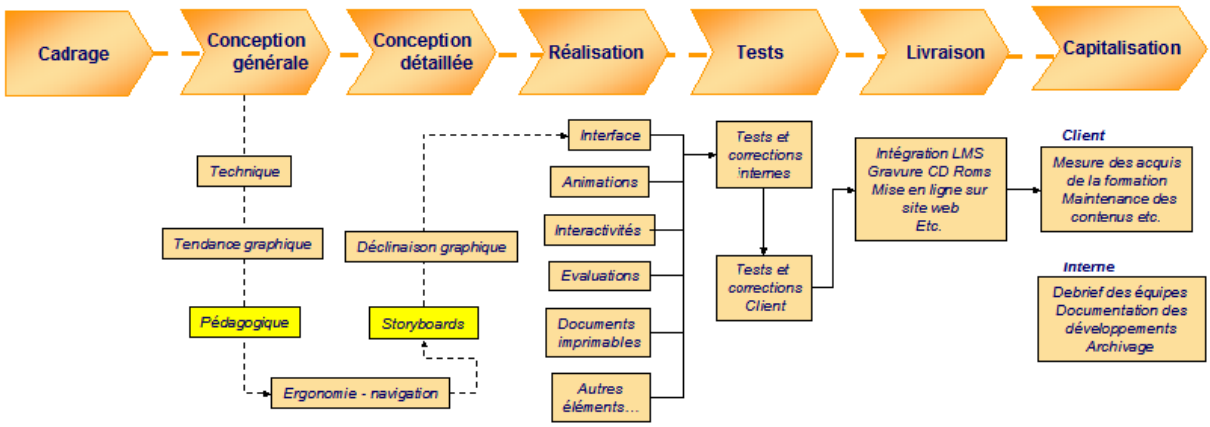
D'après ce modèle, le processus de Serious Game Design se compose de cinq étapes :

- *[A] Analyse des besoins :* définition des caractéristiques techniques et pédagogiques du Serious Game. Cette étape vise à identifier le *« contenu pédagogique »* qui doit être transmis à travers le jeu, et à le formaliser. Pour cela, les concepteurs de ***KTM Advance*** proposent d'utiliser tout d'abord une *« note de cadrage »* qui résume les besoins pédagogiques exprimés par le client. Ensuite, une analyse plus poussée du contenu à transmettre est menée *(par exemple grâce à des entretiens avec les spécialistes du sujet à traiter)*, et les différents éléments de ce contenu sont regroupés dans une liste hiérarchique.
- *[B] Proposition de Game Design :* à partir des éléments regroupés lors de l'étape précédente, le concepteur invente des mécanismes de jeux permettant de transmettre le contenu. Cette proposition de Game Design est formalisée dans un document idoine, qui détaille les divers aspects du jeu tels que : *scénario, gameplay, listes des tableaux de jeu, interactions de jeu…* Ce document rejoint le *« Game Design Document »* utilisé dans l'industrie du jeu vidéo *(p.44)*.
- *[C] Développement du prototype :* à partir des éléments inventés lors de la phase de Game Design, les programmeurs réalisent un prototype du jeu. Il permet de tester la pertinence des mécanismes de jeux proposés en regard des objectifs pédagogiques définit lors de la première étape.
- *[D] Livraisons itératives :* le jeu est ensuite développé selon un processus itératif, tel que nous avons déjà pu l'observer dans les modèles dédiés au Game Design *(p.60)*. Les deux auteurs précisent ici qu'ils optent pour des cycles d'itérations courts, dans lequel le jeu est régulièrement soumis au client, qui renvoie ensuite une liste de modifications à apporter.
- *[E] Livraison finale :* une fois les cycles d'itérations terminés, ce processus abouti à un Serious Game complet, dont la pertinence par rapport aux objectifs visés est assurée par les tests utilisateurs continus effectués durant l'étape *[D]*.

Bien que ***KTM Advance*** soit une société issue de la sphère de l'e-learning, nous pouvons voir que leur processus de conception d'un Serious Game se rapproche des modèles formalisant le processus de conception de jeux vidéo de divertissement que nous avons étudiés *(p.60)*.

2.3 LES MODELES INDUSTRIELS DE PARASCHOOL

Un autre modèle industriel nous est exposé par un des concepteurs de la société ***Paraschool***, spécialisée dans la conception de Serious Games à vocation pédagogique (Cheruette, 2009).



33. Modèle industriel de création d'un Serious Game utilisé par *Paraschool*

Le travail de conception théorique à proprement parler est ici divisé en trois étapes. La première, *« Cadrage »*, renvoie au travail préliminaire du concepteur : *déterminer la cible du projet, identification des objectifs pédagogiques, et recherche documentaire sur le sujet à traiter*. Vient ensuite la *« Conception générale »* qui vise à esquisser les grandes lignes d'un principe de jeu permettant de transmettre le contenu pédagogique : *choix d'un genre vidéoludique, de l'univers du jeu, du but du jeu, des personnages…* Cette première ébauche est ensuite détaillée lors de la phase *« Conception détaillée »* par la création de toutes les règles de jeu, de l'interface… Dans son guide méthodologique, ***Cheruette*** propose de s'appuyer sur des outils théoriques tels que nos *« briques de GamePlay » (p.267)* lors de cette étape. Le Serious Game ainsi conçu est ensuite réalisé puis testé en interne, jusqu'à sa validation finale par le client. Bien que ce modèle découpe de manière plus fine les phases de conception théorique, il propose une série d'étapes qui n'est pas sans rappeler les modèles exposés précédemment.

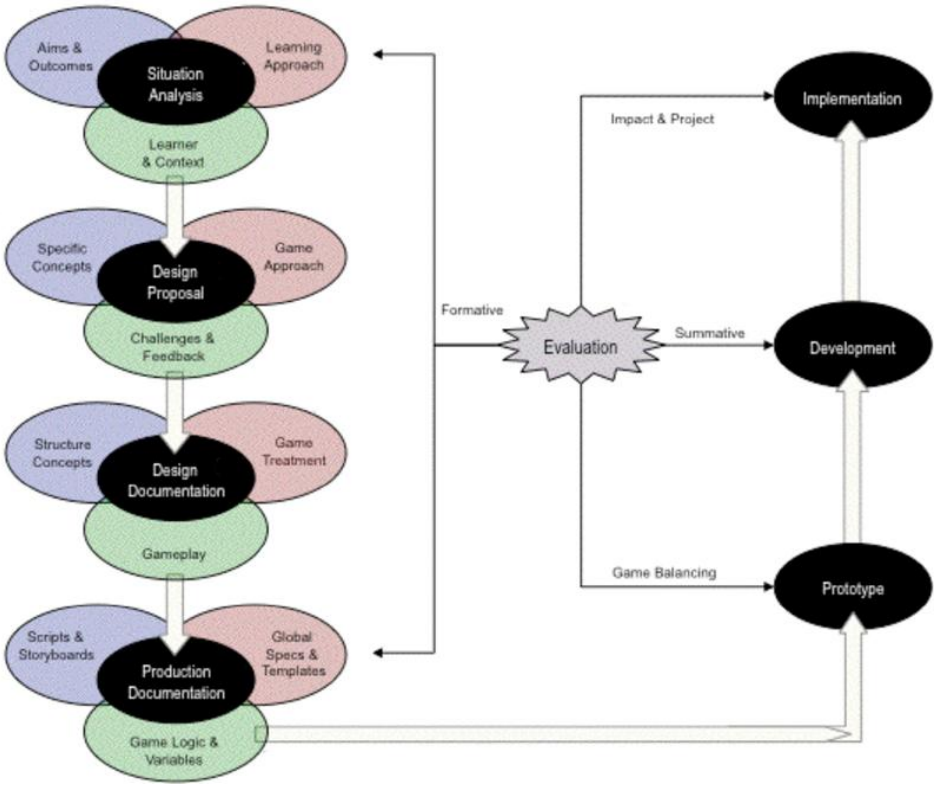
Ancienne directrice de production chez ***Paraschool***, ***Lhuillier*** (2011) propose quant à elle deux modèles du processus de conception. Son premier modèle, de type « en cascade », reprend globalement les mêmes étapes que celles proposées par ***Cheruette***. Par contre, son second modèle, inspiré par les méthodes agiles de développement, repose sur un processus itératif articulé en trois étapes :

- *Cadrage* : définition des objectifs du projet grâce à l'utilisation d'outils théoriques traditionnels *(modèles d'analyse stratégique, fiches d'interview…)*.
- *Conception* :
    - *Pédagogique* : spécification du contenu pédagogique du jeu
    - *Technique* : définition des contraintes techniques du projet
    - *Game Design* : rédaction d'un *« Game Design Document » (p.44)*
- *Prototype(s)* : partie itérative du processus dans laquelle des prototypes sont régulièrement produits *(tous les 15 jours)* et remis au client pour évaluation.
- *Livraison* : le jeu terminé est remis au client

Ces deux modèles du processus de conception d'un jeu pédagogique issus de la société ***Paraschool*** illustrent l'influence des méthodes d'ingénierie informatique *(en cascade ou agile)* sur la conception de Serious Games. Mais les sources d'inspiration ne s'arrêtent pas à ce seul domaine. Dans son ouvrage, ***Lhuillier*** propose par exemple d'utiliser de nombreux « outils théoriques », provenant de domaines allant de l'analyse marketing à la stratégie industrielle, pour assister le concepteur durant la phase *« Cadrage »*.

2.4 LE MODELE DODDEL

L'ingénierie pédagogique semble avoir inspirée plusieurs approches du processus de conception d'un Serious Game. Un de ses vecteurs d'influence est le modèle générique *ADDIE*, que nous avons déjà évoqué *(p.65)*. Certains acteurs de l'industrie du Serious Game étant issus de l'ingénierie pédagogique *(p.18)*, ils se sont inspirés de ce modèle générique pour proposer une série d'étape permettant de créer un Serious Game. Nous recensons par exemple le *modèle DODDEL* (McMahon, 2009). Acronyme de ***D**ocument-**O**riented **D**esign and **D**evelopment of **E**xperiential **L**earning*, ce dernier est composé des étapes suivantes :

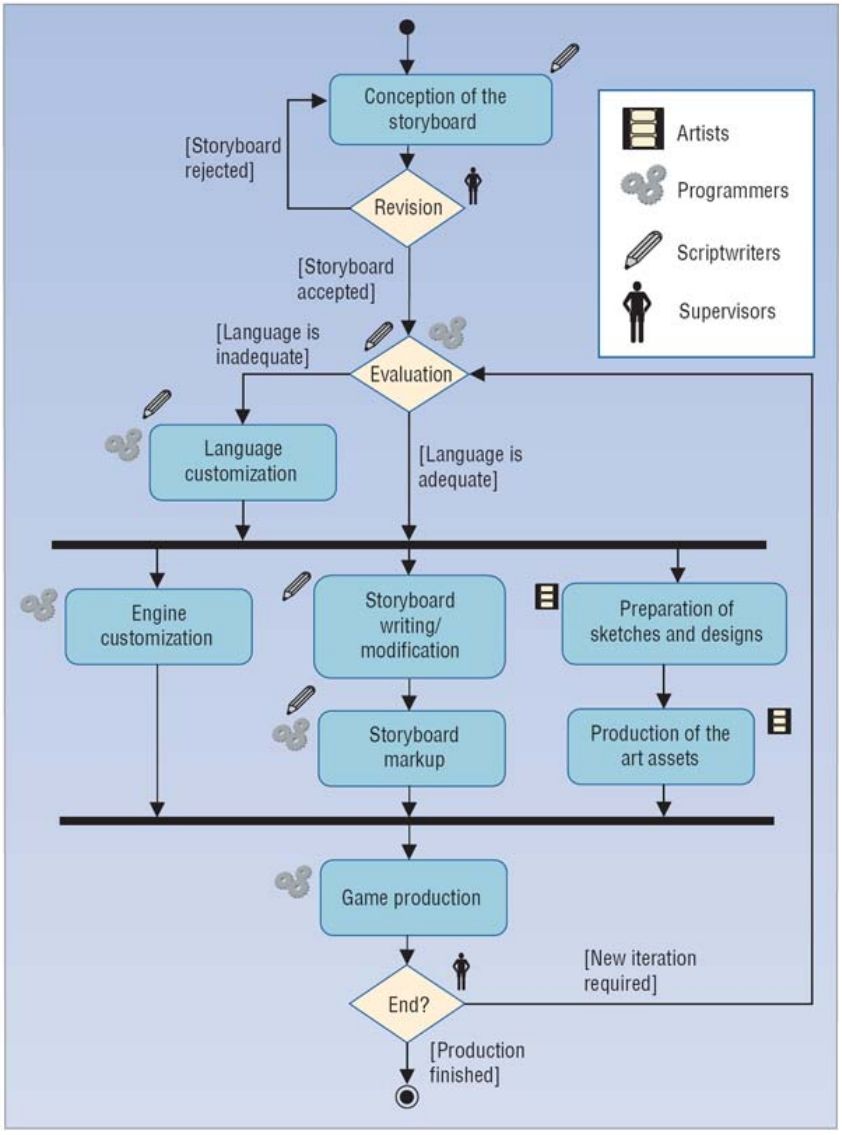


34. Le *modèle DODDEL* de *McMahon*

De l'aveu même de son auteur, le *modèle DODDEL* est basé sur le modèle générique *ADDIE*, qu'il améliore par l'introduction de boucles itératives pour certaines étapes du processus. ***McMahon*** a utilisé ce modèle pour guider des étudiants novices dans la création de Serious Games. Menée auprès d'un groupe de vingt étudiants en première année de licence de Game Design, cette expérimentation de terrain montre que le *modèle DODDEL* semble à même d'aider des novices sur deux aspects. Tout d'abord, il définit une base commune facilitant la communication interne à chaque groupe d'étudiants. Cela nous renvoie au potentiel des outils théoriques de Game Design pour la communication *(p.82)*. Mais surtout, ce modèle théorique propose une série complète d'étapes à suivre pour la création de Serious Games. Il permet ainsi de guider le processus créatif de personnes n'ayant jamais réalisé de Serious Games auparavant. Ce modèle a été utilisé ici pour former des novices souhaitant devenir de futurs concepteurs professionnels *(p.82)*. Cet exemple illustre donc le potentiel des outils théoriques pour rendre le Serious Game Design accessible à un public non spécialisé dans le domaine.

2.5 UN MODELE CENTRE SUR LE CONTENU

Une approche radicalement différente de la conception de Serious Games nous est proposée par l'équipe de chercheurs à l'origine de l'outil technique *<e-Adventure> (p.222)*. Ils partent du principe que la réalisation de Serious Games réunit au moins deux types de compétences : des experts du domaine « sérieux », et des développeurs capable de construire techniquement le jeu. Selon eux, si les experts du domaine ne sont pas encadrés, le projet se heurtera à des limitations techniques. En revanche, si les programmeurs définissent en premier lieu des limites techniques, la créativité des experts du domaine se trouvera lourdement limitée. Les chercheurs proposent alors une troisième voie, sous la forme d'un modèle *« centré sur le contenu »* (Moreno-Ger, Martínez-Ortiz, José Luis Sierra, & Fernández-Manjón, 2008). Au lieu de s'inscrire dans un cadre générique, ils se focalisent volontairement sur un genre de jeux vidéo permettant, selon eux, de diffuser du contenu éducatif de manière pertinente : le jeu d'aventure graphique *(p.174)*. Les chercheurs proposent ensuite une série d'étapes permettant la création d'un Serious Game basé sur ce genre vidéoludique :



35. Le modèle théorique de conception accompagnant <*e-Adventure*>

Ce modèle s'inspire de la méthode de conception logicielle dite du *Processus Unifié*, et plus particulièrement du *Rational Unified Process*, son incarnation la plus connue. Le modèle proposé repose donc entièrement sur un cycle itératif, qui impose aux outils techniques d'évoluer en même temps que la conception du Serious Game. Concrètement, un *« langage »* est défini au départ. Ce *« langage »* est composé par les différents éléments caractérisant un jeu d'aventure : *avatar du joueur, inventaire, notion de scène, de dialogues à choix multiples…* Ce *« langage »* est alors doté d'une représentation formelle qui permet aux concepteurs de consigner leurs idées dans un document respectant un formalisme précis. Ce formalisme est tout simplement un ensemble de balises *XML* renvoyant à chacun des éléments du *« langage »*. Une fois que les concepteurs ont consigné leurs idées dans un document en utilisant le *« langage »*, un outil technique est utilisé pour transformer ce document en un jeu fonctionnel. Cette étape s'accompagne de la création des éléments graphiques et sonores utilisés pour la représentation du jeu. Toutes ces étapes représentent un seul cycle d'itération de ce modèle, qui propose d'en utiliser plusieurs pour aboutir à un Serious Game finalisé. Cependant, comme dans la méthode *Rational Unified Process*, chaque cycle d'itération de cet outil théorique n'abouti pas forcément à un produit utilisable. A l'inverse des autres modèles utilisant des cycles itératifs, chaque cycle permet ici d'avancer progressivement dans la réalisation sans pour autant arriver à un jeu terminé à la fin d'un cycle. Un cycle se termine dès qu'il est nécessaire d'évaluer si les outils techniques permettent de réaliser ce que les concepteurs ont imaginé. Cette différence permet de s'assurer que le *« langage »*, et donc que les outils techniques utilisés pour réaliser le Serious Game, évoluent en même temps que les

idées des concepteurs, et ne constituent donc pas un frein à leur créativité. Pour autant, le fait que les idées des concepteurs soient évaluées par les développeurs avant de faire évoluer les outils permet de s'assurer de la faisabilité technique du projet. En s'inspirant d'une méthodologie issue de l'ingénierie logicielle, les chercheurs ont réussi à créer un modèle de conception de Serious Games permettant d'assurer un certain équilibre entre créativité et faisabilité. Nous reviendrons d'ailleurs sur l'outil technique qui accompagne cet outil théorique dans la troisième partie *(p.222)*.

2.6 UN MODELE POUR LA CONCEPTION DE JEUX VIDEO EDUCATIFS

Au-delà de ces approches focalisées sur les Serious Games, nous trouvons le modèle de ***St-Pierre*** (2006), qui s'inscrit dans le cadre plus général de production multimédia. Le chercheur décrit tout d'abord les différentes phases du processus de production d'un projet multimédia :

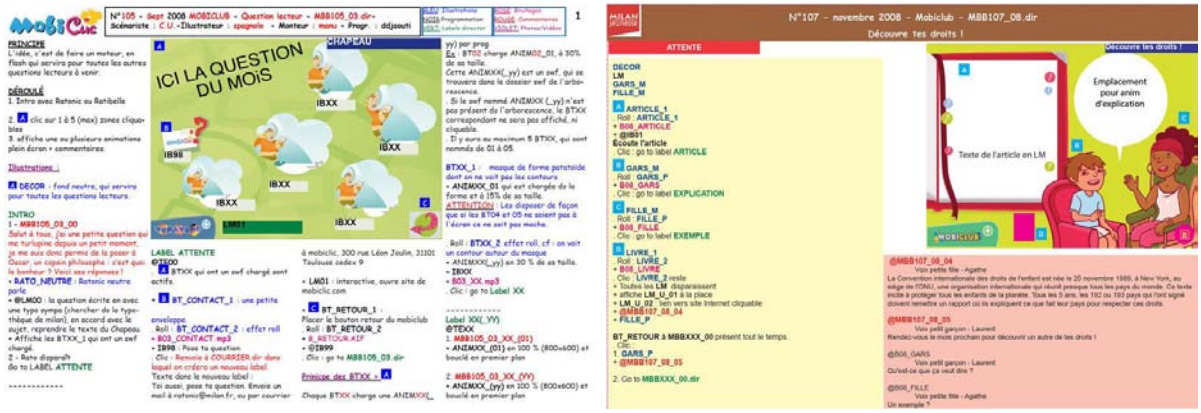
- Analyse et conception
    - *Analyse des besoins :* identification et définition des objectifs visés par le produit
    - *Développement créatif :* élaboration d'un scénario multimédia
    - *Documentation :* rédaction d'un « devis de conception » décrivant le fonctionnement du produit de manière à permettre sa fabrication
- Production et validation
    - *Production des éléments médiatiques :* réalisation des éléments graphiques, sonores… du produit
    - *Intégration – programmation :* réalisation logicielle d'un prototype fonctionnel du produit
    - *Evaluation :* tests du produit auprès d'utilisateurs, puis correction du produit selon un processus itératif
- Diffusion et maintenance

Si l'on excepte la dernière phase qui concerne la distribution d'un produit multimédia, les différentes étapes comprises dans les deux premières phases font plus ou moins écho aux modèles de conception de jeux vidéo et de Serious Games évoqués jusqu'ici. Et effectivement, ***St-Pierre*** ne propose pas de modèle du processus de production spécifique au Serious Game, car il considère ce dernier comme un projet multimédia. Il propose donc de s'appuyer sur ce modèle générique pour sa conception. Cependant, l'auteur propose quand même un outil théorique spécifique au jeu vidéo éducatif, en la forme d'un plan pour le *« devis de conception »* :

- *Intentions :* définitions des objectifs pédagogiques visés par le jeu, du public ciblé et des choix technologiques
- *Synopsis :* description générale de l'application selon deux axes : l'axe ludique et l'axe pédagogique
- *Recherche compétitive :* comparaison et analyse d'au moins trois projets similaires. Cette analyse souligne les points forts et les points faibles des jeux existants, afin de mettre en valeur la nouveauté du projet
- *Fonctionnalités particulières :* descriptions détaillées des mécanismes d'interactivité, de la façon dont le joueur va interagir avec le jeu
- *Maquettage de l'interface graphique :* illustration des écrans du projet, annotés de manière à préciser le fonctionnement des différentes zones de l'interface
- *Schématisation de l'information et des processus :* ensemble de schémas formalisant les autres aspects du projet (arborescence de navigation…) qui sont utilisés pour communiquer au sein de l'équipe, par exemple entre les concepteurs et les programmeurs, ou entre le concepteur et l'équipe commerciale chargé de la promotion du produit.

- *Fiche technique :* document illustré présentant le produit en une seule page, de manière à permettre sa promotion lors de salons, expositions…

Loin d'être purement théorique, ce modèle semble faire écho aux pratiques du secteur du multimédia éducatif. Avant d'effectuer le travail de recherche consigné dans cette thèse, nous avons eu l'occasion de travailler en tant que développeur-concepteur au sein de sociétés spécialisées dans la création de jeux dits « ludo-éducatifs » *(p.14)*, telles que la société ***Magelis*** (*Charivari de Chat-Malo : Pacha*, *La Cité Romaine*…) et le pôle multimédia de ***Milan Presse*** (*Mobilic*, *Toboclic*, *Alm@nak*...). Lors de notre participation aux jeux « ludo-éducatifs » réalisés par ces entreprises, nous avons pu observer la mise en place de méthodologies de conception similaires. Plus précisément, les concepteurs de ces sociétés avaient pour mission de rédiger un « cahier des charges » reprenant la plupart des points évoqués par ***St-Pierre*** dans son *« devis de conception »*. Par exemple, s'agissant des jeux pédagogiques se trouvant sur le magazine multimédia *Mobiclic* édité par ***Milan Presse***, le cahier des charges était composé d'une page par *« écran »* composant le jeu. Chaque page comprenait une maquette du visuel de « l'écran », sur laquelle chaque élément interactif était annoté. Ces annotations précisaient les actions qui doivent se dérouler lorsque l'utilisateur clique à cet endroit : *lancement d'une animation, d'un message d'accompagnement...*



36. Exemples de cahier des charges utilisés pour la conception de jeux pour le magazine *Mobiclic* (2008)

Au final, ce modèle, proposé par un concepteur multimédia indépendant dans un cadre universitaire, semble viser une portée plus générale que les autres. Au-delà du rappel des différentes étapes de conception d'un Serious Game pédagogique, l'auteur présente de nombreux « conseils de conception » pour chacune de ces étapes. Cette approche renvoie à celles étudiés précédemment pour le secteur du Game Design *(p.80)*

2.7 UNE APPROCHE CENTREE SUR L'UTILISATION D'UN OUTIL TECHNIQUE

Les outils techniques peuvent également être au centre d'une méthodologie théorique de conception d'un Serious Game, comme l'ont proposé les chercheurs à l'origine des outils *Adventure Author* et *Flip* *(p.225)*. En synthèse à l'étude de travaux portant sur le processus créatif, pour des domaines allant du Game Design à l'écriture en passant par l'ingénierie logicielle, ils proposent une « série d'étapes » pour la création de jeux vidéo en contexte pédagogique (Robertson & Nicholson, 2007) :

- *Exploration :* le concepteur prend connaissance de l'outil technique qu'il va utiliser, évalue ses capacités de création et commence à réfléchir en quoi elles abondent dans le sens du message qu'il souhaite transmette
- *Idea Generation :* dans cette phase, le concepteur propose de nombreuses idées et évalue leur pertinence. Il sélectionnera alors un ensemble d'idées convergentes, qui permettront d'esquisser les bases d'un concept de jeu. Durant cette phase, le

concepteur sera régulièrement amené à revenir à la phase précédente pour évaluer la faisabilité technique de son idée.

- *Game Design :* le concepteur développe les idées qu'il a retenu en un concept de jeu détaillé : mécanismes de jeu, personnages, scénario, contenu des niveaux…
- *Game Implementation :* le concepteur utilise l'outil technique pour réaliser le jeu qu'il a imaginé. Cette étape est itérative : le concepteur teste régulièrement son jeu et le corrige, quitte à modifier ce qui a été définit durant l'étape précédente. Cette étape continue jusqu'à ce qu'un jeu « complet » soit réalisé
- *Game Testing :* le concepteur teste alors son jeu en intégralité, afin d'identifier des problèmes généraux tels que l'équilibrage du jeu ou des bugs techniques.
- *Evaluation :* le concepteur fait tester son jeu par une personne issue du public ciblé. Il observe alors ses réactions, prend note de ses éventuels compliments et critiques, et peut décider de revenir aux étapes précédentes du processus pour améliorer son jeu.

Les chercheurs ont par la suite éprouvé (et confirmé) la validité de ce modèle grâce à l'observation d'enfants utilisant leurs outils techniques pour la création de jeux vidéo. Une première vague d'expérimentations a été menée en la forme d'ateliers estivaux optionnels durant une semaine. Des adolescents âgés de 12 à 16 ans ont alors pu y créer des jeux vidéo de manière libre, sans forcément intégrer une dimension « sérieuse » à leur projets. Une seconde série d'expérimentations de terrain se déroula en contexte scolaire, auprès d'enfants âgés de 10 ans. Les ateliers duraient cette fois-ci une heure par semaine, et s'intégraient à l'emploi du temps de l'école. A l'image des travaux de ***Kafai*** (1994) ou du projet *Gameplay* des écoles parisiennes *(p.206)*, ces expériences montrent l'intérêt pédagogique de la création de jeux vidéo en contexte scolaire. Au lieu de se limiter à l'utilisation d'un jeu comme support pédagogique, c'est le processus de création en lui-même qui devient alors un vecteur pédagogique *(p.235)*.

Ce modèle théorique a également servi de base pour la création d'outils techniques destinés à renforcer la créativité des ses utilisateurs, en proposant des outils adaptés aux différentes étapes. Par exemple, le choix de se spécialiser dans le genre des jeux d'aventure graphique vise à faciliter l'intégration de contenu pédagogique. Mais contrairement aux usines à jeux spécialisées dans ce genre vidéoludique *(p.174)*, les outils inventés par ces chercheurs proposent une interface pour noter et regrouper les diverses idées de la phase *« Idea Generation »*. La phase *« Game Implementation »* se trouve facilitée par le travail sur l'accessibilité de l'outil technique, tandis que la phase *« Evaluation »* s'appuiera sur une fonctionnalité d'analyse automatique des jeux intégrée au logiciel. Nous reviendrons ultérieurement sur les deux outils techniques associés à ce modèle théorique *(p.225)*.

2.8 LA METHODOLOGIE DU JEU-CADRE POUR L'EDUCATION

Une autre approche intimement liée à un outil technique est développée à travers les *« Coquilles Génériques de Jeux Educatifs (CGJE) »*. La création de Serious Games éducatifs obéit ici à une logique particulière afin de simplifier le processus de création : *« la méthodologie du jeu-cadre »* (Sauvé, 2010a). Cette approche consiste à prendre un jeu existant, et à le vider de son contenu *(« informations véhiculées par le jeu »)* pour ne garder que sa structure *(« manière de jouer »)*. Le *« jeu-cadre »* ainsi obtenu peut alors être enrichi par un nouveau contenu, notamment un contenu pédagogique. Nous reviendrons ultérieurement sur l'outil technique lié à ce projet *(p.227)*, mais pouvons déjà nous arrêter sur le processus de conception utilisé pour réaliser les *« jeu-cadre »* permettant la conception de Serious Games éducatifs. D'après ***Sauvé*** (2010a), il se compose de cinq étapes :

- *Analyse préliminaires ou planification de la CGJE :* identification du contenu pédagogique à transmettre et choix d'un *« jeu-cadre »* dont la structure va être utilisée.

- *Conception de la CGJE :* description détaillée des composants du *« jeu-cadre »* à conserver ou à modifier au sein d'un *« devis de conception »*. Ce devis de conception est composé d'illustrations des différents écrans de jeu, annotées de manière à détailler les différentes interactions possibles (Sauvé, 2010b). Cette approche n'est pas sans rappeler les *« devis de conception »* utilisés pour les projets « ludo-éducatifs » *(p.96)*.
- *Médiatisation de la CGE :* programmation des différentes fonctionnalités de la coquille pour aboutir à une première version fonctionnelle.
- *Validation de la CGJE :* évaluation de la coquille générique auprès du public de concepteurs ciblé, et révisions du logiciel selon un processus itératif si besoin.
- *Evaluation formative de jeux créés à l'aide de la CGJE :* création d'un jeu vidéo éducatif à l'aide de la coquille, et évaluation de ce jeu auprès du public d'utilisateurs ciblé.

Le processus décrit ici ne porte donc pas directement sur la création d'un Serious Game, mais sur la création d'un outil technique visant à faciliter la création d'un Serious Game éducatif. Plus précisément, il s'agit de concevoir la partie « ludique » d'un jeu éducatif, pour permettre à des enseignants de se focaliser uniquement sur la partie « sérieuse », leur permettant ainsi de créer simplement et en un temps record des jeux vidéo éducatifs. Nous étudierons donc plus en détail cet outil dans le chapitre suivant *(p.227)*.

2.9 AIDE A LA REFLEXION CREATIVE : LES SERIOUS GAME DESIGN PATTERNS

Les modèles théoriques étudiés jusqu'ici sont des méthodologies de conception. Nous avons également identifié deux outils théoriques d'aide à la réflexion créative. Le premier d'entre eux s'inscrit dans la catégorie des « pièces à assembler » pour créer un Serious Game. Nous avons déjà évoqué des approches similaires dans le domaine du Game Design, à l'image des *Game Design Patterns* *(p.79)*. Une équipe de chercheurs français s'est appuyée sur ces travaux pour en proposer une variante spécifique aux Serious Games : les *Serious Game Design Patterns* (Huynh-Kim-Bang & Labat, 2010).

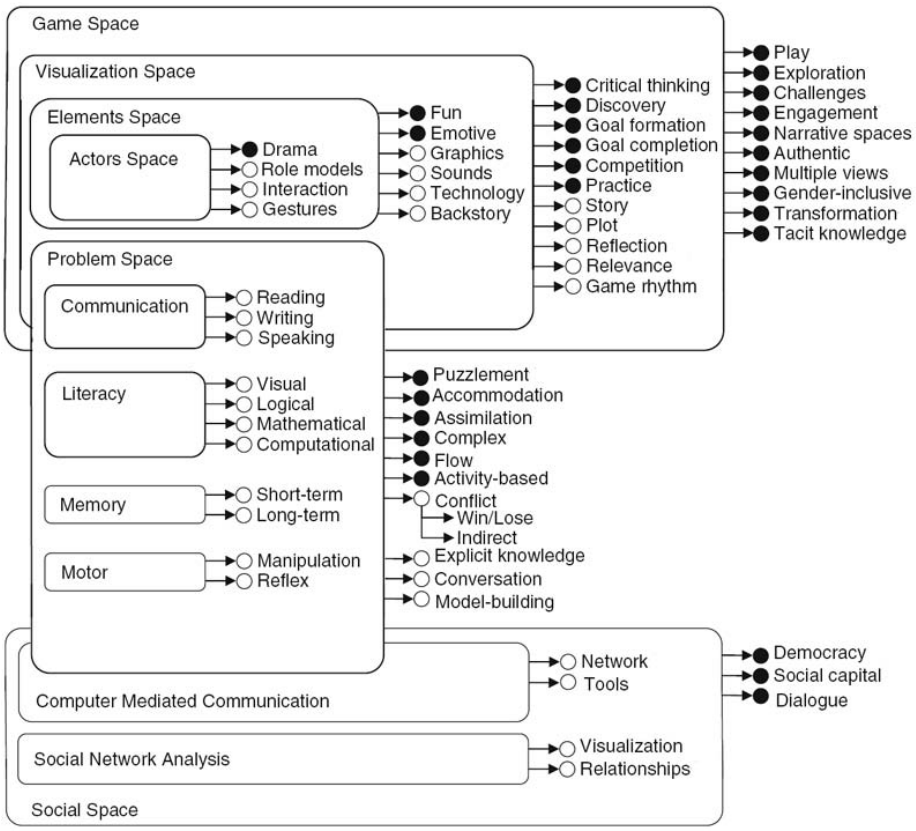
Il s'agit globalement de conseils rédigés de manière à répondre à des problématiques de conception concrètes telles que *« Comment assurer une progression continue à l'apprenant ? »* ou *« Comment encourager la réflexion ? »*. Les solutions proposées sont soit des *« patterns »* générales, telles que *« Alterner des phases d'entraînement et des phases de réflexion »* ou plus spécifiques comme *« Proposer une représentation des connaissances à obtenir sous formes d'objets virtuels à collectionner »*. Chaque solution est alors illustrée par un exemple de Serious Game afin d'être plus facilement utilisable pour les concepteurs. Au total, 37 patterns réparties en six catégories *(correspondant à des grandes problématiques de conception)* sont proposées (Huynh-Kim-Bang, 2010). Elles sont issues de l'analyse d'un corpus d'une vingtaine de Serious Games extraits de *Serious Game Classification* *(p.32)*.

Ces *Serious Game Design Patterns* peuvent être utilisés en complément des autres outils que nous venons d'évoquer. Lors des phases « d'invention de concept de jeu » décrites par les autres modèles, elles peuvent par exemple permettre d'évaluer la pertinence des idées du concepteur, ou encore lui suggérer des idées « clé-en-main » éprouvées par d'autres titres.

2.10 AIDE A LA REFLEXION CREATIVE : LE GAME OBJECT MODEL

Nous avons également pu identifier un outil d'aide à la réflexion créative basé sur un modèle formel du « jeu ». Le *Game Object Model* dresse une typologie des éléments permettant de créer un jeu éducatif (Amory, 2007). Son approche et sa finalité sont donc similaires à celle de la *typologie multidimensionnelle des jeux* *(p.68)*, mais pour le secteur du Serious Game.

Concrètement, le *Game Object Model* s'inspire du paradigme de programmation *« Orienté Objet »* pour proposer une typologie hiérarchisée des éléments constitutifs d'un jeu vidéo éducatif. Ce modèle définit des « objets », tels que *« Espace de jeu »* ou *« Espace social »*. Ces objets peuvent être soit indépendants, soit contenu les uns dans les autres. Dans ce dernier cas, un objet contenu « hérite » des caractéristiques de l'objet qui l'englobe. Ces caractéristiques sont matérialisées par des « interfaces » telles que *« Graphismes »*, *« Fun »* ou *« Esprit critique »*. Deux types d'interfaces existent. D'un côté, les interfaces « abstraites » correspondent à des notions telles que *« Fun »* ou *« Découverte »*, et sont représentées par des cercles noirs dans le schéma ci-dessous. Les notions « concrètes » telles que *« Graphismes »* ou *« Gestes »* sont quant à elles matérialisées par des points blancs. L'auteur du modèle précise que les *« interfaces abstraites »* renvoient à des notions qui seront utilisées par le concepteur du jeu, alors que les *« interfaces concrètes »* seront plutôt mises en place par le programmeur. Au final, ces interfaces représentent les différentes composantes d'un Serious Game. Le recours à une hiérarchie d'objets permet alors de les organiser.



37. La seconde version du *Game Object Model*

La première version du *Game Object Model* fut introduite en 1999. En 2007, une seconde version de cet outil théorique, enrichie de nouveaux *« objets »* et de leurs *« interfaces »*, fut proposée. La principale différence entre ce modèle typologique et ceux étudiés dans le champ du Game Design vient de l'introduction d'un objet *« Espace des problèmes »*, qui amène de nombreuses notions liées à l'acquisition de compétences scolaires comme la maîtrise de la lecture ou des mathématiques. Les autres notions proposées dans ce modèle, telles que *« Fun »*, *« Histoire »* ou *« Exploration »* se rapprochent de celles de la *typologie multidimensionnelle des jeux* *(p.68)*. Ainsi, nous pourrions dire que cet outil théorique dédié au Serious Game se distingue par l'ajout de notions liées à la dimension « sérieuse », là où les modèles similaires mais destinés au jeu vidéo de divertissement regroupent uniquement des notions liées à la dimension « ludique ».

3 SYNTHESE : FORMALISER LE PROCESSUS DE « SERIOUS GAME DESIGN » ?

A l'image de ce que nous avions observé pour les modèles du processus de Game Design *(p.65)*, il semble exister plusieurs « séries d'étapes » permettant d'aboutir à la création d'un Serious Game. La formalisation du processus de Serious Game Design à travers une « série d'étapes » qui serait universelle n'est donc pas envisageable, pas plus qu'elle ne l'était pour le processus du Game Design. Cependant, au sein de ces différents outils théoriques, nous pouvons identifier des étapes récurrentes, ainsi que des notions transversales comme le recours à des cycles de création itératifs. Nous pouvons alors envisager d'utiliser une démarche similaire à celle utilisée pour analyser les différentes méthodologies de conception de jeux vidéo de divertissement *(p.65)*. À défaut d'une série d'étapes universelle pour la conception de Serious Games, nous proposons de définir un « modèle générique ». Il s'agit d'une proposition « d'étapes génériques » destinée à inspirer les concepteurs de Serious Games pour définir leur propre série d'étapes. Pour cela, nous avions comparé et synthétisé les séries d'étapes des différents outils théoriques de notre corpus. En effectuant cette démarche avec les modèles du processus de Game Design, nous avions abouti à la définition du *modèle générique ICE* *(p.65)*.

Nous pouvons tenter de réaliser un travail similaire pour le champ du Serious Game, en essayant cette fois d'analyser et de synthétiser les séries d'étapes proposées par les dix modèles du processus de Serious Game Design étudiées dans ce chapitre. Ce faisant, nous remarquons tout d'abord que les trois étapes du *modèle générique ICE*, à savoir *« Imaginer »*, *« Créer »* et *« Evaluer »*, permettent de regrouper la plupart des étapes de ces modèles. Cependant, certaines de ces étapes ne rentrent dans aucune de ces trois « étapes génériques ». Par exemple, l'étape *« Cadrage »* des modèles de ***Cheruette*** et ***Lhuillier*** *(p.94)*, l'étape *« Analyse des besoins »* des modèles de ***KTM Advance*** *(p.93)* et de ***St-Pierre*** *(p.98)* ou encore l'étape *« Définition du contenu sérieux »* utilisée dans le processus de conception de *geDriver* *(p.86)* ne peuvent être regroupés au sein du *modèle générique ICE*. Pourtant, toutes ces étapes semblent toutes être centrées sur la même tâche : l'identification et la formalisation du contenu « sérieux » qui devra être diffusé à travers le jeu. Nous proposons alors de créer une nouvelle « étape générique » pour regrouper toutes ces étapes : *« Définir »*. Le modèle générique que nous proposons d'utiliser pour la conception de Serious Game est donc composé de quatre étapes :

- **Définir :** spécification du contenu sérieux qui devra être transmis à travers le jeu (objectifs pédagogiques, listes de connaissances à transmettre, message publicitaire…)
- **Imaginer :** à partir du contenu sérieux, le créateur invente un concept de jeu. Cette étape va généralement de pair avec l'emploi d'outils théoriques.
- **Créer :** un prototype est réalisé pour tester la pertinence de ce concept de jeu. Cette étape est généralement appuyée par l'utilisation d'outils techniques.
- **Evaluer :** le prototype est évalué auprès d'un public cible. Les critères d'évaluation varient selon les projets, mais, pour la plupart des Serious Games, la transmission effective du contenu définit lors de la première phase sera généralement mesurée.

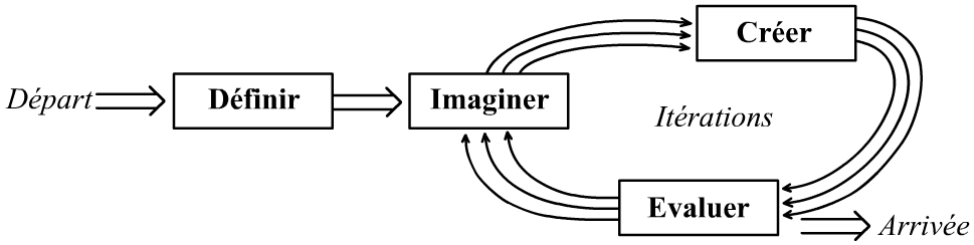
Nous proposons d'appeler ce modèle générique de conception de Serious Games, le *modèle générique DICE*. Le tableau ci-dessous expose comment les quatre étapes de ce modèle générique permettent de regrouper les séries d'étapes des dix modèles du processus de conception de Serious Game étudiés dans ce chapitre :

Tableau 4. Comparaison des modèles du processus de Serious Game Design à travers le *modèle générique DICE*

Outil théorique	Définir	Imaginer	Créer	Evaluer
Processus de conception de *geDriver (p.86)*	- Définition du contenu sérieux	- Game Design - Level Design	- Réalisation de prototypes	- Evaluation des prototypes
Modèle de ***Marfisi-Schottman & al.*** *(p.91)*	- Client's need	- Conception - Pedagogical quality control	- Production - Coherence control & debugging	- Test on test group - Use & maintenance
Modèle de ***KTM Advance*** *(p.93)*	- Analyse des besoins	- Proposition de Game Design	- Développement du Prototype	- Livraisons itératives - Livraisons finale
Modèle de ***Cheruette*** *(p.94)*	- Cadrage	- Conception générale - Conception détaillée	- Réalisation	- Tests - Livraison - Capitalisation
Modèle de ***Lhuillier*** *(p.94)*	- Cadrage	- Conception	- Prototype(s)	- Livraison
Modèle de ***McMahon*** *(p.95)*	- Situation analysis	- Design proposal - Design documentation	- Production documentation - Prototype - Development - Implementation	- Evaluation
Modèle de ***Moreno-Ger & al.*** *(p.96)*	*Conception assurée directement par les experts du domaine « sérieux »*	- Conception of the storyboard - Language customization - Storyboard writing/modification - Preparation of sketches and designs	- Engine customization - Storyboard markup - Production of the art assets - Game production	*Evaluation continue tout au long du processus (au début et à la fin de chaque cycle d'itération)*
Modèle de ***St-Pierre*** *(p.98)*	- Analyse des besoins	-Développement créatif - Documentation	- Production des éléments médiatiques - Intégration et programmation	- Evaluation - Diffusion et maintenance
Modèle de ***Sauvé*** *(p.100)*	- Analyse préliminaires ou planification de la CGJE	- Conception de la CGJE	- Médiatisation de la CGJE	- Validation de la CGJE - Evaluation formative de jeux crées à l'aide de la CGJE
Modèle de ***Robertson & Nicholson*** *(p.99)*	- Exploration	- Idea Generation - Game Design	- Game Implementation	- Game Testing - Evaluation

Nous constatons ici que chacun des modèles étudiés propose une ou plusieurs étapes pouvant être regroupée à travers notre modèle générique. A l'image de notre réflexion sur les outils théoriques de Game Design qui se concluait par les propos de ***Crawford*** *(p.84)*, nous pensons que le processus de conception d'un Serious Game reflète une part artistique, et qu'il n'est donc pas possible de le formaliser de manière universelle. Nous proposons alors d'utiliser le *modèle générique DICE* comme un canevas destiné à inspirer chaque concepteur de Serious Game dans la recherche de la « série d'étapes » qui lui convient le mieux, ou qui est la plus adaptée au contexte du projet de Serious Game qu'il doit créer. A l'image de son prédécesseur, le *modèle générique DICE* repose sur un cycle itératif. Plus précisément, la première étape de ce modèle n'intervient qu'une seule fois durant le processus de conception,

alors que les trois étapes suivantes font partie d'un cycle itératif qui commence par l'étape *« Imaginer »* pour se terminer après l'étape *« Evaluer »*, tel qu'illustré par le schéma ci-dessous :



38. Le *modèle générique DICE* du processus de Serious Game Design

En comparant le *modèle générique ICE* *(p.65)* et le *modèle générique DICE*, nous pouvons affirmer que le processus de conception de Serious Game se distingue de celui de conception d'un jeu vidéo de divertissement par la présence d'une catégorie d'étape supplémentaire : l'étape *« Définir »*. Au-delà du processus de conception en lui-même, nous pouvons également voir à travers les travaux exposés dans ce chapitre que la conception d'un Serious Game est rarement un travail solitaire, et implique donc le recours à des documents permettant aux différents acteurs de communiquer. A travers la notion de *« devis de conception »*, nous retrouvons des approches similaires au *« Game Design Document »* utilisé dans l'industrie du jeu vidéo de divertissement *(p.44)*. Bien que l'industrie du Serious Game soit plus jeune que celle du jeu vidéo, nous pouvons imaginer qu'elle pourrait elle aussi bénéficier de l'utilisation d'outils théoriques d'aide à la réflexion créative pour la communication entre les différents acteurs d'un projet *(p.82)*. De tels modèles spécifiques aux Serious Games étant plutôt rares *(p.101)*, le fait de s'inspirer de ceux issus du jeu vidéo de divertissement *(p.68)* pourrait sûrement permettre de créer de nouveaux « outils théoriques » adaptés au Serious Game. Pour cela, il convient au préalable d'essayer d'identifier de manière plus détaillée les différences entre la conception d'un jeu vidéo de divertissement et d'un Serious Game.

SPECIFICITES DU GAME DESIGN DE SERIOUS GAMES

Bien qu'il n'existe visiblement pas de méthodologie universellement applicable à la conception de Serious Games, nous avons pu identifier à travers le *modèle générique DICE* un cadre permettant d'analyser et synthétiser les différentes méthodologies que nous avons étudiées. Ce modèle se rapproche du *modèle générique ICE*, qui avait été utilisé pour synthétiser les méthodologies de Game Design *(p.65)*. Nous remarquons cependant que les processus de conception de Serious Games se distinguent de ceux du jeu vidéo par la mise en place d'une « catégorie d'étape » supplémentaire. Mais les différences entre la création d'un Serious Game et d'un jeu vidéo de divertissement ne s'arrêtent pas là. Il nous semble empiriquement que la création de mécanismes et d'univers de jeu destinés à servir des finalités « sérieuses » invite le concepteur de Serious Game à se poser des questions légèrement différentes de celles d'un Game Designer visant le seul divertissement.

Nous proposons alors d'étudier de manière qualitative quelques spécificités de la conception d'un Serious Game. Nous exposerons tout d'abord des considérations théoriques sur le sujet, avant d'analyser la genèse de deux Serious Games réalisés dans le cadre du *CIFRE* accompagnant cette thèse.

1 ASSOCIER LES DIMENSIONS « SERIEUSE » ET « LUDIQUE »

Avant de revenir en détail sur des expériences de conception de Serious Games, nous proposons d'étudier des travaux théoriques existants sur le sujet.

1.1 DIFFERENTES APPROCHES DE LA CONCEPTION DE SERIOUS GAME

Un tour d'horizon de la littérature scientifique consacrée aux Serious Games fait ressortir plusieurs approches possibles de leur conception. Plus précisément, ces différentes approches semblent liées à la manière dont les concepteurs de Serious Games essaient d'associer les dimensions « sérieuse » et « ludique » qui les caractérisent *(p.18)*.

Nous pouvons tout d'abord nous référer aux travaux de ***Egenfeldt-Nielsen*** (2006), consacrés à l'analyse de Serious Games à vocation pédagogique pour les domaines de l'éducation et de la santé. A travers un article de synthèse, ce chercheur détaille l'influence de différentes théories cognitives sur la conception des jeux vidéo pédagogiques. Son analyse met en évidence un lien entre chaque théorie et une manière particulière de mélanger les dimensions « ludique » et « pédagogique ». Par exemple, il observe que des titres s'inscrivant dans le courant *« behavioriste »* séparent explicitement la dimension pédagogique de la dimension ludique, en utilisant cette dernière pour récompenser l'utilisateur qui réussit la partie pédagogique. Le plaisir de jeu est ici utilisé comme un facteur de motivation *« extrinsèque »* pour l'apprentissage d'un contenu sérieux. A l'inverse, les titres s'appuyant sur les théories *« constructivistes »* mélangent ces deux dimensions de manière à ce qu'il ne soit plus possible de les distinguer. La motivation de l'apprenant découle donc ici de facteurs *« intrinsèques »*, le contenu sérieux étant au cœur du principe de jeu. S'il s'agit là des deux principales théories cognitives que ***Egenfeldt-Nielsen*** identifie comme approches de conception d'un Serious Game, ce chercheur cite également une troisième théorie utilisée pour l'apprentissage par le

jeu vidéo : le *« constructionnisme »*. Cette théorie partage la nature *« intrinsèque »* du *« constructivisme »*, mais s'articule autour d'un processus de création comme vecteur d'apprentissage. Le *« constructionnisme »* ne s'appuie donc pas directement sur l'utilisation d'un Serious Game pour l'apprentissage, mais permet de mettre leur création au cœur d'une activité pédagogique. Nous reviendrons ultérieurement sur cette approche différente mais très intéressante *(p.235)*.

Les constatations de ***Egenfeldt-Nielsen*** ne semblent pas être réservées aux seuls secteurs de la santé et de l'éducation. En effet, ***Chen & Ringel*** (2001) aboutissent à des conclusions similaires suite à une étude des Serious Games à caractère publicitaire. Ainsi, ils distinguent trois types de stratégies publicitaires mobilisant le jeu vidéo :

- *Associatif :* utilisation d'une structure ludique sur laquelle est appliquée un contenu publicitaire détaché du contexte du jeu. *Exemple : mise en place de panneaux publicitaires dans le jeu Sportura the game* ***(Nonoche & Medialand, 2004)***. *En plus des publicités décorant le circuit de ce jeu de course, le temps restant est affiché sur un chronomètre arborant une marque célèbre.*
- *Illustratif :* utilisation d'une structure ludique sur laquelle est appliquée un contenu publicitaire lié au contexte du jeu. *Exemple : utilisation des différentes générations de voitures comme « buzzer » permettant de répondre à un quiz musical dans Volkswagen Drive In* ***(Achtung!, 2007)***. *Dans ce titre, le joueur doit retrouver la « décennie » de sortie d'une chanson en cliquant sur le modèle de voiture correspondant.*
- *Démonstratif :* intégration du message publicitaire au cœur même de la structure ludique, de façon à ce qu'il ne soit plus possible de les considérer séparément. *Exemple : mise en avant de certains modèles de voitures dans Volvo S60 Concept* ***(Simbin, 2009)***. *Dans ce titre, le joueur peut directement conduire le modèle de véhicule qui est valorisé par le Serious Game.*

Si nous recoupons ces stratégies publicitaires avec les approches observées pour les jeux vidéo pédagogiques, nous remarquons que les stratégies *Associatives* et *Illustratives* reposent toutes deux sur une juxtaposition des dimensions « sérieuse » et « ludique », renvoyant donc à l'approche *« extrinsèque »* influencée par les théories béhavioristes. A l'inverse, la stratégie *Démonstrative*, qui mélange les deux dimensions, se rapproche clairement de l'approche *« intrinsèque »* et des théories constructivistes. Comme nous l'avons déjà évoqué, l'éducation, la santé et la publicité représentent les principaux secteurs du Serious Game *(p.35)*. D'un point de vue quantitatif, les observations de ces deux études nous semblent donc pertinentes pour l'ensemble du secteur. En conclusion, nous identifions plusieurs manières de concevoir un Serious Game, qui semblent s'inspirer de deux grandes approches :

- **L'approche « extrinsèque »**, dans laquelle les dimensions « sérieuse » et « ludique » sont séparées. Un concepteur de Serious Games s'inscrivant dans cette approche aura tendance à alterner des phases de jeu avec des phases de transmission du contenu sérieux. Le jeu est alors utilisé comme une « récompense » suite à la compréhension du contenu sérieux.

- **L'approche « intrinsèque »**, qui vise à mélanger les dimensions « sérieuse » et « ludique » de manière à ce qu'il ne soit plus possible de les séparer. Un concepteur de Serious Game s'appuyant sur cette approche cherchera à intégrer le contenu sérieux au sein des mécanismes de jeu. La réussite au jeu découle donc directement de la compréhension du contenu sérieux.

Bien que présentant chacune leurs avantages et leurs inconvénients, ces deux approches n'ont visiblement pas suscité le même intérêt pour les concepteurs de Serious Games. Ainsi, comme le note ***Egenfeldt-Nielsen*** (2006), l'approche « extrinsèque » fut particulièrement prisée durant les années 1980 et 1990, se retrouvant au cœur de la plupart des ancêtres des Serious Games actuels, courant du « ludo-éducatif » en tête *(p.33)*. Si ces titres sont utilisés avec succès par des parents ou enseignants à des fins d'apprentissage, les Serious Games basés sur l'approche « extrinsèque » souffrent des mêmes critiques qui sont adressées aux théories béhavioristes.

Par exemple, le fait que l'apprentissage soit lié à un facteur de motivation externe au contenu « sérieux » n'est pas sans poser problème. Ainsi, l'efficacité du jeu *TechnoCity* ***(Rectorat de Toulouse, 2006)***, basé sur une approche « extrinsèque », a été longuement étudiée (J. Alvarez, 2007). Ce titre est destiné à sensibiliser des collégiens en 3^{e} à différents métiers industriels et techniques, tels qu'électricien, chef de chantier ou mécanicien. Pour chacun des cinq métiers mis en avant par le jeu, la démarche est identique. Tout d'abord, le joueur se voit proposer un jeu sur le thème du métier. Ainsi, pour le métier d'électricien, un jeu de plateforme demande au joueur d'incarner un personnage devant récupérer des lucioles pour alimenter une centrale électrique. Après deux niveaux de ce jeu rappelant *Super Mario Bros.* ***(Nintendo, 1985)***, un court reportage vidéo illustrant le métier d'électricien est projeté. Ces quelques minutes d'interviews de professionnels du métier se concluent par un questionnaire à choix multiple. La question posée porte sur le contenu du reportage vidéo. Si le joueur répond correctement à la question, il peut passer à la suite du jeu, et ainsi parcourir deux niveaux supplémentaires du jeu de plateforme avant de retomber sur une nouvelle vidéo assortie de questions. S'il répond mal, il doit choisir une autre réponse, en visionnant éventuellement le reportage à nouveau. Nous voyons clairement que ce jeu propose d'un côté des phases de contenu « sérieux », à travers les vidéos d'information, et des phases « ludiques » récompensant un joueur qui a prêté attention au contenu « sérieux ». Bien que les concepteurs aient pensé les jeux de la partie « ludique » par rapport aux compétences du métier illustré *(par exemple, un jeu de plateforme nécessite de l'agilité comme le métier d'électricien)*, l'évaluation de ce Serious Game sur le terrain est on ne peut plus mitigée. Ainsi une étude menée par ***Alvarez*** dans un collège fait ressortir les conclusions suivantes :

> *La majorité de ces collégiens consultent souvent Technocity essentiellement pour jouer. […] Pour les garçons qui prétendent ne pas tenir compte des vidéos, lorsque nous leur demandons ce qu'ils retiennent avec Technocity, les réponses sont clairement associées aux jeux pratiqués. Pour la fille de 12 ans, qui consulte les vidéos dans le but de répondre au quiz, un amalgame s'opère entre les jeux et les vidéos. […] La collégienne semble faire une confusion avec le jeu dédié à la filière "Ingénierie mécanique" proposant à l'utilisateur d'assembler des scooters. Un jeu qu'elle affectionne particulièrement. Quant aux collégiens qui disent avoir consulté les vidéos, les informations restituées sont assez laconiques voire confuses. [p.88]*

Comme le note ***Alvarez*** dans sa thèse, le fait que les dimensions « sérieuse » et « ludique » de *TechnoCity* soient présentées de manière alternées aux joueurs n'a pas véritablement aidé ce jeu à délivrer son message. La plupart des collégiens se sont uniquement consacrés aux phases de jeu proposées comme « récompense », délaissant leur poste lors des phases de présentation d'une vidéo. Ils allaient alors aider leurs camarades à compléter les phases de jeu, puis revenaient répondre aux questions posées par les vidéos. Soit les collégiens s'échangeaient entre eux les bonnes réponses pour aller directement à la phase de jeu, soit ils répondaient au hasard jusqu'à trouver la bonne réponse. Si certains collégiens ont néanmoins été sensibles au message délivré par ce Serious Game, allant même jusqu'à se renseigner sur

un métier donné, pour la plupart de ses utilisateurs le constat est plutôt mitigé. S'il cela n'est pas uniquement lié à l'approche « extrinsèque » choisie par ce titre, l'étude de terrain fait ressortir qu'il s'agit d'un des facteurs majeurs du manque de pertinence observé. Notons d'ailleurs que cette même tension entre « jeu » et « apprentissage » dans les Serious Games « extrinsèque » fut également étudiée et critiquée en des termes similaires pour les titres du courant du « ludo-éducatif » (Kellner, 2000, 2007).

Contrairement à l'approche « extrinsèque », l'approche « intrinsèque » est relativement peu répandue dans les ancêtres des Serious Games. Quelques rares titres comme *Packy & Marlon* ***(Raya Systems, 1994)*** et *Bronkie the Bonchiasaurus* ***(Raya Systems, 1994)***, respectivement destinés à l'apprentissage des précautions de vie liées au diabète et à l'asthme, ont montré des résultats très encourageants auprès d'enfants atteints de ces maladies (Brown et al., 1997; Lieberman, 2001). De même, des travaux de recherche ont montré le potentiel de l'approche « intrinsèque » dès le début des années 1980 (Malone, 1981). Mais ces quelques initiatives n'ont pas véritablement fait écho dans les Serious Games des années 1980 et 1990 dominés par l'approche « extrinsèque » (Egenfeldt-Nielsen, 2006). Cette tendance semble néanmoins s'inverser avec la vague actuelle des Serious Games *(p.37)*. En effet, les études récentes sur la conception de Serious Games semblent favoriser l'approche « intrinsèque ». Que ce soit du côté de l'industrie ou de la recherche, les concepteurs de Serious Games semblent donc avoir assimilés les critiques adressées à l'approche « extrinsèque », et expérimentent actuellement l'approche « intrinsèque ». Par exemple, pour la société ***KTM Advance***, les approches de type « extrinsèque », influencées par les théories béhavioristes, se trouvent au cœur de leurs produits de type *« e-learning »*. En revanche, leur gamme de Serious Games s'inscrit dans une approche « intrinsèque » afin d'explorer d'autres modes de transmission pédagogique (Boudier & Dambach, 2010). Au-delà des aspects pédagogiques, cette approche semble présenter pour la société un réel avantage lors de l'évaluation des apprenants :

> *« Un premier indicateur fiable de succès [des apprenants] est le passage des différents niveaux du jeu : si le Game Design a été correctement conçu, les objectifs pédagogiques sont encapsulés dans les différents éléments de gameplay qui permettent de progresser et de passer un à un les niveaux. [p.162] »*

Ainsi, l'approche « intrinsèque » permettrait de faciliter l'évaluation de l'impact d'un Serious Game en suivant le parcours du joueur. Si nous reviendrons ultérieurement sur cet aspect *(p.120, p.126 et p.145)*, nous notons ici le caractère ouvertement « intrinsèque » de l'approche de conception de Serious Games utilisée par cette société. Du côté de la recherche, nous pouvons par exemple nous référer aux travaux de ***Bruckman*** (1999), qui critique ouvertement l'approche « extrinsèque ». Elle remarque tout d'abord que cette approche correspond à des titres cherchant à rendre attrayant l'apprentissage de contenus grâce à des « pauses ludiques » faisant office de récompenses. Suite à l'analyse de ces titres, elle conclut que cette approche « extrinsèque » est aussi vaine que celle de *« recouvrir un brocoli de chocolat »* pour que des enfants l'apprécient. Elle appelle donc les concepteurs de jeux pédagogiques à expérimenter d'autres approches où l'apprentissage est au cœur de l'activité ludique *(cet article est antérieur à la vague actuelle des Serious Games)*. L'appel lancé par ***Bruckman*** trouve, entre autres, un écho dans les travaux de ***Habgood*** (2007) sur la pertinence de l'approche « intrinsèque » pour la création de jeux vidéo pédagogiques. Dans sa thèse, ce chercheur et concepteur de Serious Games développe et évalue un jeu vidéo destiné à l'apprentissage des mathématiques pour des enfants de 7-8 ans : *Zombie Division*. Plusieurs versions de ce Serious Game ont été créées par le chercheur en s'appuyant soit sur une approche « extrinsèque », soit sur une approche « intrinsèque ». Une version du jeu « débarrassée de tout contenu éducatif » fut également utilisée comme témoin lors des expérimentations de terrain. Ces expérimentations font tout d'abord ressortir que les approches « extrinsèque » et « intrinsèque » sont plus ou moins équivalentes en terme de

motivation générale des élèves. Par contre, sur la version « extrinsèque », divisée en sessions de « jeu » alternées avec des « quiz mathématiques », les élèves ont émis les mêmes remarques que pour *TechnoCity*. Ils ont étés gênés par *« l'ennui »* procuré par ce quiz, auquel il ne prêtaient d'ailleurs que peu d'attention car ils étaient pressés de *« continuer à jouer »*. A l'inverse, la version « intrinsèque » du Serious Game a été saluée pour son côté *« amusant et fun »*. Cette dernière version semble également avoir permis un apprentissage de meilleure qualité, en permettant aux élèves d'apprendre plus de contenu et de le retenir plus longtemps qu'avec la version « extrinsèque ».

Si plusieurs facteurs rentrent bien évidemment en ligne de compte, le fait que les élèves n'aient plus de « rupture » dans leur pratique du jeu et soient donc exposés simultanément aux dimensions « ludique » et « sérieuse » semble ici avoir un effet positif sur l'impact du Serious Game. Ainsi, un Serious Game « intrinsèque » semble donc en mesure de susciter le *« flow »* chez les utilisateurs. Cette notion, introduite par le psychologue ***Csikszentmihalyi*** (1990), correspond à un état mental dans lequel l'utilisateur se consacre totalement à la réalisation d'une tâche donnée. Pour cela, cette tâche doit posséder certaines caractéristiques, telles que le fait d'être adaptée aux compétences de l'utilisateur, de lui donner des retours qualitatifs régulier sur son travail et de posséder des objectifs clairs. Cet état de concentration maximale s'accompagne généralement d'une distorsion de la notion de temps et permet d'aboutir à un sentiment de réussite une fois la tâche accomplie. Si l'état de *« flow »* peut être suscité par de nombreuses activités comme la lecture, la peinture, le sport ou la musique, nous le retrouvons également à travers le jeu vidéo. Dans l'industrie du jeu vidéo de divertissement, certains Game Designer s'appuient par exemple sur cette théorie afin de concevoir des titres particulièrement immersifs et prenants (Jenova Chen, 2006). Mais nous voyons ici qu'elle peut être également intéressante pour le Serious Game, et que l'approche « intrinsèque » permet justement de conserver le *« flow »* durant toute la partie, là où l'approche « extrinsèque » est obligée de le briser pour diffuser son contenu sérieux.

Quel que soit son intérêt en terme de motivation et d'immersion, le *« flow »* n'est pourtant pas une recette miracle pour l'apprentissage. Par exemple, si nous revenons sur l'expérimentation de ***Habgood*** (2007), nous constatons que la bonne qualité d'apprentissage observée sur le terrain n'est pas uniquement liée au Serious Game en lui-même. En effet, les meilleurs résultats d'évaluation avec ce jeu ont été obtenus lors de l'utilisation de la version « intrinsèque » conjuguée à une séance de discussion menée par l'instituteur. Après une session de jeu, l'enseignant a joué un rôle essentiel pour permettre à ses élèves de prendre conscience du contenu pédagogique auquel ils ont été confrontés en les faisant réfléchir sur les mécanismes du jeu auquel ils venaient de jouer. Ainsi, lorsqu'un joueur est complètement immergé dans la sensation de *« flow »* que lui procure un Serious Game « intrinsèque », la problématique est d'arriver à amener le joueur à prendre du recul sur son expérience ludique pour mieux assimiler le contenu « sérieux » auquel il a été confronté. A ces fins, certains concepteurs de Serious Games n'hésitent pas à créer des jeux faisant volontairement perdre le joueur pour l'inciter à réfléchir sur les mécanismes de jeux, et donc sur le contenu qu'ils véhiculent (Lee, 2003). Nous reviendrons plus en détail sur cette méthode permettant d'allier *« flow »* et « prise de recul » lors de notre retour d'expérience *(p.132)*. En attendant, nous constatons à travers ces quelques exemples que si l'approche « extrinsèque » était plébiscitée par les ancêtres des Serious Games, les titres de la vague actuelle semblent principalement s'appuyer sur l'approche « intrinsèque », et commencent donc à en explorer les nombreuses possibilités. Nous essaierons modestement d'en faire de même à travers une expérimentation de terrain *(p.115)*.

Si l'approche « intrinsèque » permet de diffuser un contenu sérieux à l'intérieur d'un jeu vidéo, cela ne signifie t'il pas simplement que tout jeu vidéo est potentiellement un vecteur d'apprentissage ?

1.3.1 LES « PRINCIPES D'APPRENTISSAGE » DU JEU VIDEO DE DIVERTISSEMENT

Nous avons déjà observé qu'à travers la pratique du *« Serious Gaming »*, il était possible d'utiliser certains jeux vidéo de divertissement à des fins utilitaires *(p.22)*. Cela signifie donc que ces titres, qui ne sont pas pensés pour un usage utilitaire, transmettent néanmoins quelque chose au joueur. Cela est-il pour autant réservé à certains jeux vidéo ? Des travaux tels que ceux de ***Prensky*** (2007), ***Gee*** (2007) ou ***Schaffer*** (2007) nous laissent au contraire penser que tous les jeux vidéo impliquent un apprentissage. Ces trois chercheurs en sciences cognitives ont séparément menés des recherches sur le potentiel du jeu vidéo, et de tous les jeux vidéo, pour l'apprentissage. Si ***Schaffer*** expose comment les jeux vidéo permettent d'aborder les problèmes à travers différents points de vue, ***Prensky*** examine comment la culture vidéoludique et la manière dont les jeux vidéo sont construits peuvent être réinvestis pour améliorer l'éducation. De son côté, ***Gee*** tient les propos suivants:

> *« Nous avons ici des jeux vidéo de divertissement qui sont longs et difficiles à terminer. Cependant, vous ne pouvez pas jouer à un jeu sans l'apprendre. Et si personne ne joue à un jeu donné, il ne se vend pas et la société qui l'a créé fait faillite. Donc les concepteurs de tels jeux pourraient raccourcir et simplifier leurs titres pour en faciliter l'apprentissage. C'est souvent ce que font les écoles avec les cursus scolaires. Mais les joueurs de jeux vidéo ne veulent pas de titres courts et faciles. Alors, les Game Designers continuent à produire des jeux longs et complexes, mais arrivent néanmoins à ce que les joueurs apprennent à y jouer. Comment ? […] Il serait particulièrement intéressant d'enquêter sur les principes d'apprentissage qu'ils déploient. Comment les bons jeux vidéo sont-ils conçus pour faciliter leur apprentissage – un apprentissage de bonne qualité pour que les gens puissent y jouer et les apprécier même si le jeu est long et difficile ? Ce que nous cherchons donc à identifier est la théorie d'apprentissage humain intégrée aux bons jeux vidéo.*[34] *»*

Ainsi, ***Gee*** analyse les jeux vidéo de divertissement et arrive en à en déduire 36 *« principes d'apprentissage »* tel que *« le principe des solutions multiples »* ou *« le principe de l'incrémentation »*. Ces 36 *« principes d'apprentissage »* issus du jeu vidéo de divertissement sont ainsi formalisés afin d'inciter les pédagogues à s'en inspirer pour leurs cursus scolaires. Les travaux de ces trois chercheurs en sciences cognitives rejoignent les conclusions de Game Designers professionnels tels que ***Koster***, pour qui le plaisir que procurent les jeux vidéo *(le « fun »)* est directement lié au fait que le joueur apprend en y jouant *(p.73)*.

Si tous les jeux vidéo impliquent un apprentissage, pourquoi fait-on alors une distinction entre jeu vidéo de divertissement et Serious Game ? Tout simplement parce que si tous les jeux vidéo permettrent d'apprendre, la nature de ce qu'ils transmettent n'est pas forcément

[34] *« So here we have something that is long, hard and challenging. However, you cannot play a game if you cannot learn it. If no one plays a game, it does not sell, and the company that makes it goes broke. Of course, designers could keep making the games shorter and simpler to facilitate learning. That's often what schools do with their curriculums. But gamers won't accept short and easy games. So game designers keep making long and challenging games and still manage to get them learned. How? […] It would seem intriguing, then, to investigate what these principles of learning are. How are good video games designed to enhance getting themselves learned – learned well so people can play and enjoy them even when they are long and hard? What we are really looking for here is this: the theory of human learning built into good videogames. »*

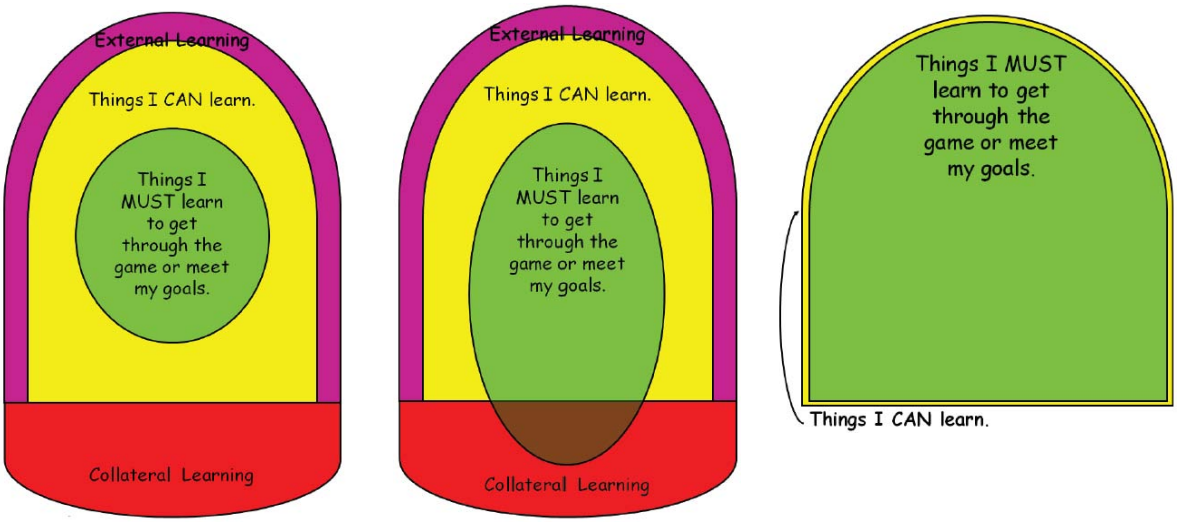
« sérieuse ». Ainsi, la plupart des jeux vidéo de divertissement imposent l'apprentissage de notions qui sont uniquement applicables pour gagner au jeu. Cette constatation est illustrée par le modèle formel de ***Cook*** *(p.73)*, qui modélise les *« compétences »* que le joueur doit acquérir afin de gagner le jeu. Ces *« compétences »* ne sont pas forcément transférables dans la vie réelle, et se limitent à mettre en place une « stratégie gagnante » liée au jeu. Dans ce cas, comment différencier Serious Game et jeu vidéo de divertissement ? Tout simplement en évaluant la nature « utilitaire » de ce qu'ils permettent d'apprendre. Si l'apprentissage se limite au seul cadre du jeu, nous restons dans la sphère du divertissement. Mais si les compétences acquises par le biais du jeu ont une application externe à ce dernier, nous pouvons considérer qu'il rentre dans le cadre du *« Serious Game »*, ou à défaut dans celui du *« Serious Gaming »* *(p.25)*. Ainsi, l'évaluation du contenu d'apprentissage d'un jeu vidéo permettrait d'appréhender sont potentiel « sérieux ». Cette approche fut développée par une chercheuse du nom de ***Becker*** pour proposer un modèle permettant d'analyser le contenu d'apprentissage véhiculé par un jeu vidéo.

1.3.2 LE MODELE MAGIC BULLET

De l'aveu même de son auteur, cet outil se destine aussi bien à l'analyse de jeux existants qu'à la conception de nouveaux titres, à l'image des modèles « typologiques » *(p.68)* que nous avons étudiés précédemment. Dénommé *Magic Bullet*, ce modèle catégorise ce que le joueur peut apprendre à travers un jeu donné selon quatre catégories (Becker, 2011) :

- *Apprentissage potentiel (jaune) :* renvoie à toutes les informations incluses dans le jeu par son concepteur, et que le joueur pourra potentiellement apprendre durant la partie.
- *Apprentissage obligatoire (vert) :* sous-ensemble de la catégorie précédente, désigne les informations incluses dans le jeu par le concepteur et que le joueur doit apprendre afin de gagner le jeu.
- *Apprentissage collatéral (rouge) :* renvoie à l'apprentissage émergent qui peut avoir lieu à l'extérieur du jeu. Par exemple, si un joueur de *God of War III* ***(SCE, 2010)*** manifeste l'envie de se renseigner sur la mythologie grecque après avoir joué à ce titre, il s'agit d'un apprentissage qui n'est pas directement lié au jeu, qui n'a pas été anticipé par son concepteur, qui ne permet pas de gagner au jeu, mais qui est quand même motivé par la pratique du jeu.
- *Apprentissage externe (violet) :* renvoie à l'apprentissage, issu de sources extérieures, qui permet de gagner au jeu. Par exemple, de nombreux joueurs produisent des *FAQs*, sortes de guides stratégiques librement diffusés par Internet *(p.200)*. Lorsqu'un joueur consulte ces guides afin de gagner au jeu, il est en situation d'apprentissage externe.

Cette catégorisation simple permet de lister les différentes formes d'apprentissage d'un jeu vidéo donné. Cette liste est ensuite convertie sous forme de schémas, permettant ainsi de quantifier rapidement les différents types d'apprentissages présents :



39. Exemples de schémas produits avec le modèle *Magic Bullet*

Grâce à de tels schémas, il est possible d'estimer le rapport entre ce que le joueur peut apprendre dans un jeu et ce qu'il doit apprendre pour le gagner, ou encore de noter la part laissée aux apprentissages externes et collatéraux. Par exemple, l'auteur indique que le schéma de gauche renvoie à des jeux au processus d'apprentissage bien équilibré, et correspond à des grands classiques du jeu vidéo de divertissement tels que la série des *The Legend of Zelda* ***(Nintendo, 1986-2009).*** Le schéma central, dans lequel un apprentissage collatéral est requis pour gagner au jeu, correspond au jeu *Where in the World is Carmen Sandiego?* ***(Brøderbund Software, 1985)***, qui demande au joueur de bien maîtriser la géographie en consultant des atlas afin de gagner. Le dernier schéma correspond quant à lui à des jeux ou tout le contenu doit être appris afin de gagner, par exemple dans un titre comme *Math Blaster* ***(Davidson & Associates, 1994)***. Dans un autre article, ***Becker*** (2007) va même jusqu'à comparer ce même *Math Blaster*, un Serious Game « extrinsèque » destiné à l'apprentissage des mathématiques, à *New Super Mario Bros.* ***(Nintendo, 2006)***, un classique du genre vidéoludique du « jeu de plateforme ». *Math Blaster* est également basé sur un jeu de plateforme, qu'il utilise comme « récompense » si le joueur répond correctement aux nombreux quiz mathématiques qui composent le jeu. D'après l'auteur, bien que l'apprentissage de ces deux jeux s'appuie sur le principe « d'essais et erreurs », leur pertinence en terme d'apprentissage est très différente. Par exemple, de part sa nature « extrinsèque », il est possible de gagner à *Math Blaster* sans forcément maîtriser les connaissances mathématiques que le jeu est censé permettre d'apprendre. En effet, l'auteur constate qu'il est tout à fait possible de choisir des réponses au hasard, et d'arriver ainsi à passer quand même les quiz sans effectuer les calculs mentaux qu'ils sont censés impliquer. Concernant *New Super Mario Bros.*, il n'est pas possible de terminer les niveaux du jeu sans maîtriser les divers mouvements du personnage. Terminer ce jeu implique donc d'apprendre le contenu qu'il souhaite transmettre *(bien que ce contenu n'ait aucune utilité apparente en dehors du contexte du jeu)*.

1.3.3 UNE APPROCHE DIFFERENTE POUR TEMPERER L'ENTHOUSIASME DE GEE

Notons cependant que ***Linderoth*** (2010) essaie de tempérer l'engouement autour du potentiel pédagogique des jeux vidéo en affirmant que, si tous les jeux impliquent un apprentissage, dans la pratique cet apprentissage peut s'avérer très limité. Il s'appuie pour cela sur la théorie de la *« psychologie écologique »*, une théorie cognitive qui s'oppose directement au béhaviorisme et au constructivisme. D'après cette théorie, l'homme n'a pas besoin de s'appuyer sur des représentations mentales pour utiliser son environnement, mais est capable d'interpréter directement les possibilités d'interaction qu'il lui offre *(notion « d'affordance »)*. ***Linderoth*** s'appuie sur cette théorie pour affirmer que les joueurs n'ont pas besoin de construire des représentations mentales pour arriver à jouer à un jeu vidéo, mais qu'ils

déduisent tout simplement des possibilités d'interaction *(affordances)* leur permettant de terminer le jeu. Ainsi, au lieu de construire de nouvelles représentations mentales, les joueurs apprennent seulement à distinguer les différents obstacles de l'univers du jeu et à réagir en appuyant sur le bouton adéquat. Cela impliquerait donc que le processus d'apprentissage d'un jeu ne permet pas d'apprendre un contenu qui soit transférable à l'extérieur du jeu. A partir de l'analyse des systèmes d'aide qui sont couramment intégrés aux jeux vidéo actuels *(mise en surbrillance des éléments à récolter, messages d'instructions contextuels…)*, ***Linderoth*** conclut que l'apprentissage des jeux vidéo est relativement limité et consiste globalement à *« savoir différencier les éléments en surbrillance de ceux qui ne le sont pas »*. Ses observations sur le fait que les systèmes d'aide intégrés au jeu réduisent l'effort cognitif demandé au joueur, sûrement au point d'empêcher parfois tout apprentissage, semblent tout à fait pertinentes. Cependant, nous identifions un biais dans l'approche de ce chercheur. En effet, l'étude de ***Linderoth*** ne semble avoir portée que sur des jeux du genre « action », et qui plus est uniquement sur des titres récents (*LEGO Indiana Jones 3*, *Batman Arkham Asylum*, *Assassin's Creed 2*…). Comme nous le constaterons ultérieurement, les différents genres vidéoludiques ne sont pas tous égaux pour l'apprentissage *(p.119)*, les jeux orientés « action » n'étant pas les plus à même de diffuser une grande quantité de contenu. De plus, si les titres actuels, poussés par le courant du *« Casual Game »* *(p.147)*, intègrent effectivement de nombreux systèmes d'aide destinés à faciliter l'utilisation du jeu, les titres plus anciens étaient dépourvus de toute forme d'aide intégrée, et nécessitent donc un effort cognitif plus important pour un joueur qui souhaite apprendre à y jouer. Au final, le recours à la psychologie écologique de ***Linderoth*** permet de mettre en évidence qu'il est possible de concevoir des jeux vidéo qui ne demandent qu'un apprentissage limité de la part du joueur. En effet, le chercheur indique avoir volontairement choisi des jeux embarquant des systèmes d'aide facilitant grandement le jeu afin de tempérer les propos de ***Gee*** qui pourraient laisser à penser que *« tous les bons jeux vidéo impliquent un apprentissage conséquent »*. La lecture croisée des travaux de ces deux chercheurs nous laisse donc supposer qu'un jeu vidéo peut tout à fait impliquer un apprentissage très important pour le joueur, à condition que le concepteur du jeu l'ait décidé ainsi.

1.4 SYNTHESE

Tous ces travaux de recherche nous renvoient finalement à la distinction opérée par ***Becker*** (2011) entre la notion *« d'éducation »*, qui implique l'acquisition de connaissances utiles à la société, et *« d'apprentissage »*, qui correspond à l'acquisition de nouvelles connaissances, quelles qu'elles soient. Selon cette distinction, tous les jeux vidéo impliquent bien un apprentissage, mais tous ces apprentissages n'ont pas forcément de vocation utilitaire. Lorsque un concepteur envisage que son jeu vidéo soit destiné à l'apprentissage d'un contenu « sérieux », alors nous sommes en présence d'un Serious Game. Lorsque l'apprentissage du jeu vidéo n'a pas d'autres finalités que celle du jeu, nous sommes en présence d'un jeu vidéo de divertissement. Mais il peut parfois arriver que certaines personnes tierces puissent trouver une application « sérieuse » au contenu appris à travers un jeu vidéo qui n'a pas été conçu pour cela. Nous sommes alors en présence d'un détournement d'usage qui caractérise le *« Serious Gaming »* *(p.25)*.

En conclusion, ces considérations théoriques nous éclairent tout d'abord sur les différentes approches possibles pour la conception d'un Serious Game. Parmi ces différentes philosophies, nous remarquons que l'approche « intrinsèque » semblent actuellement la plus prometteuse. Si elle avait été relativement peu employée jusqu'à présent, la vague actuelle de Serious Games semble être déterminée à explorer le potentiel de cette approche qui fait écho aux *« principes d'apprentissage »* que l'on retrouve dans tous les jeux vidéo. Nous proposons donc de continuer à analyser les subtilités de l'approche « intrinsèque » à travers un retour d'expérience de terrain sur la conception de plusieurs Serious Games.

2 RETOUR D'EXPERIENCE : DEUX SERIOUS GAMES POUR UN ECOQUARTIER

Le principal projet au cœur de la convention *CIFRE* ayant permis la réalisation de ce travail de recherche est la création d'un Serious Game destiné à des fins de communication et de pédagogie à l'échelle locale. La ville de ***Blagnac***, située dans l'agglomération toulousaine, héberge un des projets d'écoquartier les plus importants de France, *Andromède*. Un écoquartier est un projet d'aménagement urbain qui respecte des principes d'aménagement durable et incite ses habitants à adopter une attitude écocitoyenne. Un tel quartier est par exemple aménagé de manière à permettre la récupération des eaux de pluie, la limitation des transports par la mise à disposition de commerces de proximité, à n'accueillir que des constructions respectant des normes environnementales… S'étendant sur 210 hectares, l'écoquartier *Andromède* est traversé par la première ligne de tramway de l'agglomération toulousaine, mélange bâtiments résidentiels, bureaux et installations publiques, et possède près de 70 hectares d'espaces verts. La ***SEM Constellation***, aménageur responsable de ce projet urbain, souhaitait utiliser un mode de communication original pour mettre en avant cet écoquartier. A l'initiative de la ***ludothèque Odyssud***, domiciliée dans un centre culturel rattaché à la mairie de ***Blagnac***, un consortium s'est réunit pour étudier le potentiel du jeu vidéo en tant que support de communication pour *Andromède*. L'***Institut de Recherche en Informatique de Toulouse (IRIT)*** et la société ***OKTAL*** ont alors rejoint ce consortium afin de créer un Serious Game visant deux objectifs :

- **Présenter l'écoquartier Andromède à la population locale.** Les habitants de la région doivent être en mesure de se familiariser avec la géographie de ce nouvel espace en train d'être construit dans leur ville. La diffusion du jeu coïnciderait d'ailleurs avec l'arrivée des premiers habitants.

- **Expliquer ce qu'est un écoquartier, et détailler la manière dont Andromède a été aménagé**. En complément de l'aspect communication, l'objectif du projet est de proposer une sensibilisation au thème de l'écologie à travers une explication pédagogique du concept d'écoquartier.

Ainsi, le projet de Serious Game exposé dans cette section diffère du projet *geDriver* *(p.86)* par de nombreux aspects. Tout d'abord, la finalité du projet : ici il n'est plus question de prodiguer un entraînement mais de diffuser des messages. Ensuite, le public visé est le « grand public », au sens large. Ce Serious Game doit pouvoir s'adresser à toute personne, homme ou femme, de n'importe quel âge et quelle que soit son expérience préalable du jeu vidéo. Enfin, le budget alloué au projet est loin des 1.2 millions d'euros de *geDriver*. Nous pouvons estimer le budget global de création de jeux vidéo sur l'écoquartier *Andromède* à 70.000€, en comptabilisant le temps consacré par chaque intervenant sur le projet. Si la réalisation des éléments graphiques et sonores a été confiée à des prestataires externes, le projet a principalement mobilisé quatre acteurs en interne. Tout d'abord, la directrice de la ***ludothèque Odyssud*** a été la coordinatrice du projet. Elle a également participé activement à de nombreuses séances de réflexion sur les principes de jeu, ainsi qu'aux tests et évaluations de prototypes avec l'ensemble de l'équipe de la ludothèque. Ensuite, les deux responsables de la communication de la ***SEM Constellation*** ont joué le rôle d'experts du domaine sérieux. Elles ont également participées à la phase de conception à travers des réunions de réflexion et des nombreux tests sur les divers prototypes de jeu. Enfin, nous avons été en quelque sorte la « cheville ouvrière » de ce projet, en endossant à la fois les rôles de concepteur et de programmeur. Nous sommes donc en mesure de vous proposer un retour détaillé de la conception de ce projet de Serious Game utilisé comme outil de communication locale. D'un point de vue méthodologique, nous avons pris des notes quotidiennes sur l'avancée de ce projet. Sur les 18 mois de réalisation du projet, cela représente 98 pages de notes sur ses divers aspects : *évolutions des concepts de jeu, compte rendu des sessions de tests, journal de*

développement technique… Nous proposons ci-dessous une synthèse analytique de ce « journal de bord », en revenant chronologiquement sur les différentes étapes du projet.

2.1 DEFINITION DU CONTENU SERIEUX

En écho aux méthodologies de conception de Serious Games analysées précédemment *(p. 103)*, ce projet débuta par une phase de définition du contenu sérieux à transmettre. Plusieurs entrevues ont été organisées avec les quatre acteurs principaux du projet. Les deux responsables communication de la ***SEM Constellation*** ont défini une liste de « valeurs » représentant *Andromède* qu'elles souhaitaient diffuser à travers le jeu. Ces « valeurs » se traduisent par des informations liées à l'écoquartier comme *« les parkings des établissements publics sont mutualisés afin de limiter l'imperméabilisation des sols en contrôlant son goudronnage »*… Ces différentes « valeurs » sont organisées en thèmes tels que *« genèse du projet »*, *« optimisation des déplacements »*…

Si le projet urbain de cet écoquartier était globalement finalisé sur le papier, sa construction était toujours en cours lors de la réalisation du Serious Game. A défaut de pouvoir visiter des bâtiments qui n'existaient pas encore, de nombreux visuels d'architecture et autres plans d'aménagements ont été mis à notre disposition. Les responsables communication ont également mis en avant un outil de communication qui a été réalisé par la société ***OKTAL*** : une maquette interactive en 3D représentant le quartier *Andromède* avec tous ses bâtiments construits. Cette maquette interactive associe une modélisation 3D du quartier avec un outil de visualisation en 3D temps réel proposant une haute qualité graphique : *Nova*[35].

2.2 CREATION D'UN CONCEPT DE JEU

Une fois le contenu sérieux défini de manière précise, et tout en gardant à l'esprit les deux objectifs posés en début du projet, débute la phase de création d'un concept de jeu permettant de transmettre tout ce contenu. Il s'agit d'une phase exploratoire durant laquelle de nombreuses idées et principes de jeux sont proposés, avant d'être éventuellement validés pour poursuivre leur développement par des prototypes.

2.2.1 ANALYSE DE L'EXISTANT

La création de concepts de jeu est précédée par l'analyse de titres éventuellement existants sur un thème similaire. Pour cela, nous nous sommes appuyés sur le riche corpus de *Serious Game Classification (p.32)*, afin d'identifier d'autres Serious Games qui auraient été réalisés pour mettre en avant un quartier ou une zone urbaine. Nous avons identifié deux références.
Tour Racing ***(Virtual Dream, 2001)*** met en valeur la province sud-coréenne de ***GyeongBuk*** pour inciter les touristes à la visiter. *Velo'v Racing in Lyon* ***(Coyote Software, 2006)*** fut commandité par la ville de ***Lyon*** afin de célébrer sa « fête des lumières », et donc promouvoir l'attrait touristique de la ville. Comme leurs titres le laissent penser, ces deux Serious Games sont des jeux de course : le but est d'arriver à franchir la ligne d'arrivée le plus vite possible en doublant des concurrents, tout en évitant de percuter des murs. Notre analyse de ces deux jeux, très prenants et remplissant sûrement les objectifs de leurs commanditaires, fait ressortir la faible pertinence du principe de jeu de course pour faire découvrir la géographie de lieux. En effet, un joueur pratiquant un tel jeu n'a pas besoin d'apprendre la disposition géographique des lieux pour gagner. Au lieu de se dire *« je traverse le parc de la tête d'or »*, il va plutôt penser *« il ne faut pas que je touche cette barrière ni cet arbre en prenant ce virage à gauche »*. Au lieu de favoriser l'exploration et la découverte de lieux inconnus, un jeu de course incite donc à identifier des obstacles pour établir la meilleure trajectoire possible

[35] Plus d'information sur cette technologie à l'adresse : http://www.vertice.fr/

(p.111). Si le plaisir de jeu est bien là, la transmission du contenu sérieux, en tout cas la découverte de la géographie des lieux, est plus que limitée avec ce principe de jeu. Evidemment, cette analyse se place dans l'optique d'un Serious Game basé sur une approche de conception « intrinsèque », qui est la voie que nous souhaitons développer. Tel que nous l'avons déjà évoqué, une approche « extrinsèque » se satisferait pleinement d'un jeu dont les mécanismes ne sont pas directement liés au contenu sérieux *(p.106)*. Mais dans le cadre de ce projet, le principe de jeu de course ne semble pas pertinent. Nous avons donc du imaginer d'autres concepts de jeu permettant de diffuser notre contenu sérieux de manière « intrinsèque ».

2.2.2 CONCEPT #1 : NAVIGATION TRIDIMENSIONNELLE ET MINI-JEUX

Les sources d'inspiration pour les concepts de jeux sont aussi diverses que variées. La première d'entre elles fut la maquette interactive réalisée par ***OKTAL***. Tout d'abord, cette maquette proposait déjà une modélisation complète et précise de l'écoquartier, ce qui semblait être une bonne base pour expliquer sa géographie aux habitants de la région. Une logique économique rentre également en ligne de compte. Il fallait réaliser ce Serious Game dans un budget relativement restreint. Réutiliser des ressources existantes constitue alors un moyen de réaliser des économies. De plus, l'utilisation de cette maquette était plutôt amusante, même si la navigation dans un espace tridimensionnel avec un clavier et une souris demande un certain temps d'adaptation pour les novices. Le premier concept de jeu proposait donc de naviguer librement dans une représentation 3D du quartier, et d'associer des « mini-jeux » à chacun des bâtiments. Ce jeu serait accompagné d'une version « allégée » s'inspirant de *Mario Party 8* ***(Hudson Soft, 2007)*** : le quartier serait transformé en « jeu de l'oie » entrecoupé des « mini-jeux » développés pour accompagner la maquette 3D. L'avantage de ce concept était de proposer deux modes d'accès complémentaires à ces mini-jeux. Avec la maquette 3D, une personne seule et habituée aux jeux vidéo en 3D pourrait explorer librement le quartier, alors qu'avec la version « jeu de l'oie » plusieurs joueurs pas forcément familiers avec le jeu vidéo seraient en mesure de s'affronter dans un titre à la manipulation plus aisée. Il a également été posé le fait que les « mini-jeux » devaient reposer par essence sur une idée simple. Chaque mini-jeu ne mettrait donc en avant qu'une seule des « valeurs » définies comme contenu sérieux.

2.2.3 CONCEPT #2 : ENQUETE DANS LE QUARTIER

Un autre concept de jeu s'inspire quant à lui de *Where in the World is Carmen Sandiego?* ***(Brøderbund Software, 1985)***, un ancêtre des Serious Games très populaire pour l'apprentissage de la géographie. Après avoir longuement analysé et déconstruit les mécanismes de ce titre, nous avons trouvé son principe de « chasse au voleur » particulièrement pertinent. Le jeu est découpé en divers « lieux » représentés par les capitales des pays du monde. Dans chaque lieu se trouvent des personnages qui donnent des indices permettant de deviner le prochain lieu dans lequel il faut se rendre. Ce principe semblait prometteur pour mettre en place une motivation poussant le joueur à visiter le quartier *Andromède*. La notion de « chasse au voleur » a tout d'abord été pensée dans un contexte local *(un vol commis à la boulangerie que doit résoudre un enfant du quartier…)*, avant d'être transformée en une « chasse au trésor » perçue comme *« plus crédible et plus valorisante en terme d'image »*. Une autre idée forte de ce concept de jeu reposait sur l'enrichissement du mécanisme de dialogue avec les personnages, afin d'essayer de transmettre de nombreuses « valeurs » du contenu sérieux dans ces conversations. En résumé, ce concept propose au joueur de se rendre dans un lieu du quartier. En discutant avec des personnages se trouvant sur place, le joueur récoltera des indices lui permettant d'identifier le lieu suivant, et ainsi de suite jusqu'à atteindre la fin du parcours.

2.2.4 INFLUENCE DES CONTRAINTES TECHNIQUES

A ce stade de la réflexion, l'idée est donc de développer deux jeux complémentaires. Tout d'abord un jeu de « chasse au trésor » permettant d'appréhender la géographie du quartier, car le passage d'un lieu à l'autre est le « but » du jeu. Le second titre rassemble de nombreux mini-jeux accessibles à la fois par un « jeu de l'oie » et à travers une navigation libre au sein de la modélisation 3D du quartier. Cependant, le choix de la 3D temps réel soulève trois problèmes majeurs :

- Tout d'abord, nous avons fait tester la maquette 3D interactive à plusieurs utilisateurs novices. Il en est ressorti que sa manipulation est vraiment trop complexe pour ce public. Ne parvenant pas à prendre l'outil en main pour se rendre là où ils le souhaitent, les utilisateurs délaissent le dispositif au bout de quelques minutes.
- Ensuite, la qualité graphique et le nombre de polygones de cette grande modélisation 3D nécessitent un ordinateur très puissant pour tourner de manière optimale. Nous avons donc testé cette maquette sur de nombreuses machines au sein du centre culturel ***Odyssud***, ainsi que sur les ordinateurs des employés de la ***SEM Constellation***. La majorité de ces machines, à la configuration certes modeste, ne peuvent faire tourner le programme de manière fluide. Cela constitue un obstacle majeur, car notre Serious Game vise un public ne possédant pas forcément du matériel haut de gamme.
- Enfin, si la distribution d'un tel jeu sur cédérom ne serait pas un problème, sa diffusion par Internet serait plus compliquée. Tout d'abord, il s'agit d'une technologie qui n'est pas utilisable directement dans le navigateur. Il faut donc télécharger et installer le programme. Ensuite, une fois compressée, la maquette interactive 3D toute seule pèse déjà 190Mo. Si l'on y ajoute les nombreux éléments graphiques et sonores qui seront inévitablement produits pour transformer cette maquette en jeu, nous pouvons estimer un temps de téléchargement du jeu relativement important.

Au final, le choix de cette technologie pour de la 3D temps réel complique la distribution du Serious Game pour les novices en informatique, jouant en défaveur de sa mission de *« communication auprès du grand public »*. Ainsi, afin de répondre à un impératif de diffusion aisée et transparente permettant de toucher une large part du public, un choix technique draconien a dû être fait : l'abandon de la 3D temps réel. Nous nous sommes alors tourné vers la technologie *Flash* *(p.186)*, qui, si elle est limitée à de la 2D, permet de créer des jeux vidéo légers en poids *(de 1 à 5 Mo)* et jouables directement dans un navigateur Internet.

2.2.5 AFFINAGE DES CONCEPTS DE JEU EN REGARD DU CONTENU SERIEUX

Peu après ce choix technique, les deux idées de concept de jeu ont été affinées au travers de nombreuses réunions de réflexion. D'un côté, un « jeu d'exploration » permettra de découvrir le quartier, et de l'autre des mini-jeux permettront d'aborder sa construction. Suite à l'abandon de la maquette 3D, ces mini-jeux deviennent autonomes.

A partir de ces deux idées de concepts de jeu, nous avons alors repris la liste des « valeurs » définies comme contenu sérieux. Nous les avons analysées et réorganisées de manière à les diviser en deux catégories : d'un côté les informations liées à la vie quotidienne du quartier une fois qu'il sera terminé, et de l'autre les informations traitant de sa construction. Le premier groupe d'informations, très nombreuses, devra être transmis à travers le jeu d'exploration, alors que les quelques informations du second groupe seront le sujet d'autant de mini-jeux. Nous avons poussé cette séparation de manière à toucher un public large. En partant du principe qu'un même jeu peut difficilement contenter tous les profils de joueurs *(p.71)*, le jeu d'exploration sera destiné à un public novice. Il sera donc particulièrement simple à manipuler, afin d'être accessible à toute personne n'ayant jamais jouée à un jeu vidéo. Il sera également basé sur des principes de « réflexion » plutôt que de « réflexes ». En proposant une exploration et de nombreux dialogues, il devrait être à même de diffuser de nombreuses informations sur l'écoquartier *Andromède*. A côté de ce titre, les mini-jeux

viseront un public plus familier avec la culture vidéoludique. Ils s'orienteront vers des principes de jeu plutôt basés sur « l'action », où la rapidité d'exécution est un des moyens permettant de gagner au jeu. Ces mini-jeux sont d'envergure plus modeste par rapport au contenu sérieux, selon le principe suivant : chaque mini-jeu diffusera une seule information de notre contenu sérieux.

Ces pistes de réflexions furent validées par les représentants des « commanditaires » du projet, à savoir la ***SEM Constellation*** et la ***Mairie de Blagnac***. Le processus de création de Serious Game s'est alors poursuivi dans deux directions parallèles : au lieu de créer un seul titre, nous réaliserons plusieurs Serious Games selon deux approches complémentaires. A partir de ce stade, la réflexion sur les concepts de jeu s'est accompagnée de la réalisation de prototypes et de leur évaluation, en accord avec les quatre étapes du *modèle générique DICE (p.103)*. Cette méthodologie a permis d'amener nos deux esquisses de concept de jeu à deux Serious Games complets, dont nous présentons séparément la suite du processus de conception ci-après.

2.3 ECO-REPORTER, A LA DECOUVERTE D'ANDROMEDE

Cette section expose la conception d'un Serious Game basé sur le concept #2 *(p.117)*.

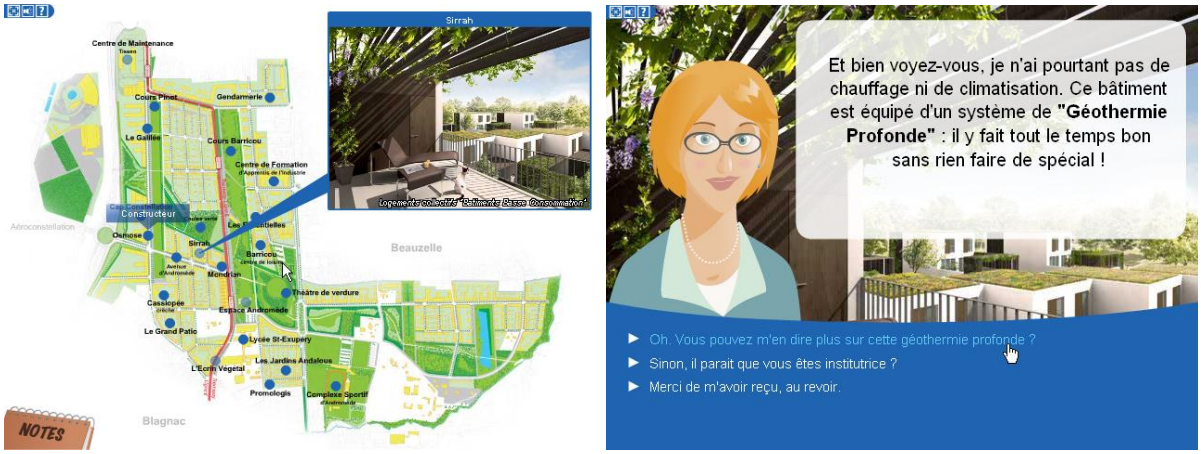
2.3.1 CHOIX D'UN GENRE VIDEOLUDIQUE

Le choix d'un genre vidéoludique est une étape cruciale lors de la conception d'un Serious Game, tel que le reflète la méthodologie déployée par ***KTM Advance*** *(p.93)*. Pour ce premier Serious Game, nous nous sommes basé sur le genre du « jeu d'aventure graphique » *(p.174)*. De par sa nature, ce genre de jeu pousse à la réflexion plutôt qu'au réflexe : le joueur est amené à explorer divers espaces, à dialoguer avec des personnages, à rassembler des objets ou encore à résoudre des énigmes. Le temps est rarement limité, chacun pouvant ainsi avancer à son rythme. De plus, la manipulation s'appuie uniquement sur l'utilisation de la souris, qui n'est utilisée que pour cliquer sur divers éléments à l'écran. Un jeu d'aventure est donc suffisamment simple pour être appréhendé par un public novice en la matière. Les concepteurs de l'outil *<e-Adventure> (p.222)* ont d'ailleurs choisi de réaliser une usine à jeu spécialisée dans ce genre vidéoludique car ils estiment qu'il est particulièrement adapté à la diffusion de contenu pédagogique (Moreno-Ger et al., 2008). Nous avons donc retenu le « jeu d'aventure » comme genre pour ce premier Serious Game, qui vise à diffuser de nombreuses informations sur l'écoquartier *Andromède*. Nous proposerons au joueur d'explorer le quartier, à travers un nombre défini de « lieux » emblématiques. Il sera également possible de discuter avec les habitants pour « comprendre » quel est le mode de vie dans un écoquartier tel qu'*Andromède*. Si nous avons ici une base permettant de mettre en scène les nombreuses informations définies dans le contenu sérieux, comment pouvons-nous, à travers le concept même du jeu, amener le joueur à s'y intéresser ?

2.3.2 INTEGRER LE CONTENU SERIEUX AU CŒUR DU PRINCIPE DE JEU

Comme nous l'avons vu lors de l'analyse de l'existant *(p.116)*, ce n'est pas parce qu'un jeu intègre des informations que le joueur va nécessairement s'y intéresser. D'une manière simple, nous sommes partis du principe que si une information « sérieuse » n'est pas utile pour « gagner » au jeu, alors le joueur ne s'y intéressera pas *(notion « d'apprentissage obligatoire » p.112)*. Ce postulat correspond totalement à la philosophie de conception « intrinsèque » *(p.108)*. Au lieu d'une chasse au trésor, nous avons alors proposé une autre approche du jeu d'aventure : endosser le rôle d'un journaliste. La mission proposée au joueur est d'écrire un article de journal sur l'écocitoyenneté. Pour cela, il se rend au cœur d'un écoquartier, en l'occurrence *Andromède*, pour y mener une enquête de terrain. Il pourra prendre en photos les différents lieux, et discuter avec les nombreux habitants. Il récupérera

ainsi des informations utilisables pour rédiger cet article. Si le joueur veut « gagner » au jeu, il doit donc partir à la recherche des différentes « valeurs » définies par le contenu sérieux, et ensuite les manipuler pour créer son propre article.



40. *Eco-Reporter* : exploration du quartier *(gauche)* et dialogue avec un personnage *(droite)*

Afin de rester accessible à un public n'ayant jamais joué à un jeu vidéo, la partie « rédaction de l'article » ne nécessitera pas d'écrire soi-même les notes. Lors des phases de dialogue, le joueur gagnera automatiquement des notes pré-écrites s'il pose les bonnes questions. Ces notes pourront ensuite être assemblées comme des pièces de puzzle pour créer la trame d'un article : il s'agira tout simplement d'organiser les notes et photos de manière linéaire. Le jeu se chargera d'adapter cette trame pour la mettre en page comme un véritable article de journal, lorsque le joueur décidera que son brouillon est « terminé ».

41. *Eco-Reporter* : rédaction d'un article *(gauche)* et article mis en forme automatiquement *(droite)*

2.3.3 EVALUER LA RECEPTION DU CONTENU SERIEUX GRACE AU PRINCIPE DE JEU

Ce concept de jeu, qui a été validé par tous les partenaires, possède un autre avantage : il intègre directement les modalités d'évaluations souhaitées par les commanditaires du jeu. En effet, une des attentes évoquées lors de la phase de définition du contenu sérieux est d'être en mesure d'évaluer l'impact des différentes « valeurs » de l'écoquartier *Andromède* auprès du public. Par exemple, les commanditaires cherchent à identifier les thèmes qui intéressent le plus le public qui sera exposé au jeu : *écologie dans la construction de bâtiment, sécurité dans le quartier, présence de commodités…*

La première piste évoquée pour cette évaluation était de faire apparaître un « quiz » en fin de partie afin de récolter les impressions des joueurs. Si l'idée n'est pas en soi inintéressante, d'autres travaux ont montré les limites d'une telle approche « extrinsèque » reposant sur la juxtaposition d'une partie « jeu » avec une partie « quiz » *(p.108)*. Avec le concept proposé

pour *Eco-Reporter*, l'évaluation de la réception du contenu sérieux par les joueurs est au cœur même du principe de jeu. En rédigeant un article, les joueurs sont amenés à choisir les informations qu'ils trouvent les plus intéressantes, et les photos des lieux qu'ils préfèrent. Il suffit alors de récolter les articles créés par les différents joueurs afin d'obtenir des statistiques sur les informations auxquelles ils ont été les plus sensibles *(p.135)*. À travers ce concept de jeu, le joueur est donc amené à dévoiler son rapport au contenu sérieux qui lui a été présenté. Le fait que cette « interrogation » soit incarnée par l'objectif même du jeu *(rédiger un article)* ne permet pas à un joueur de la considérer comme « distincte » du jeu, et donc de la négliger, s'il souhaite gagner la partie. Nous sommes donc bien dans une approche « intrinsèque ».

2.3.4 FORMALISATION DES REGLES DU JEU POUR REALISER LES PROTOTYPES

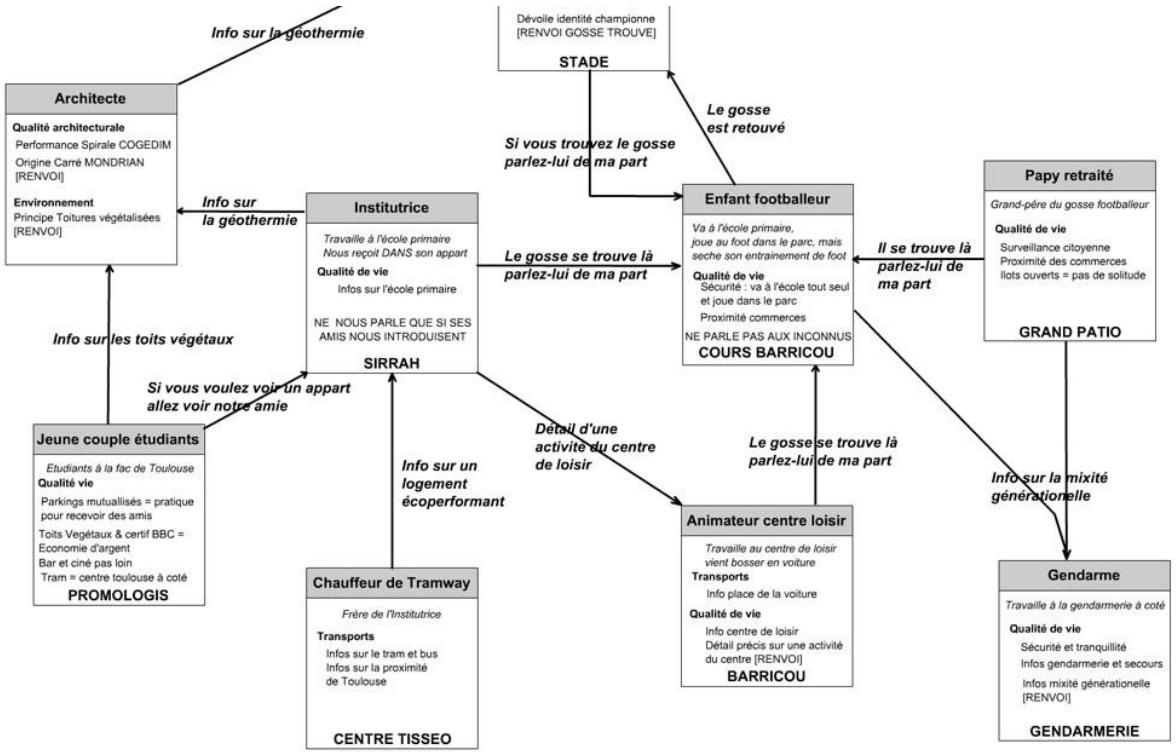
Une fois le principe de jeu clairement défini, sa réalisation s'est faite selon le processus itératif définit par le *modèle générique DICE (p.103)*. De nombreux prototypes ont été réalisés, évalués, puis corrigés jusqu'à aboutir à une version finalisée du Serious Game.

Les prototypes ont tout d'abord permis de valider la pertinence du principe de jeu. Ce faisant, l'idée théorique du fonctionnement des divers mécanismes de jeu a du être précisée de manière à être implémentée dans un programme informatique fonctionnel. Par exemple, nous avons du formaliser un système de dialogue plus concret que ce qui a été défini lors de la phase de réflexion sur le concept de jeu. Les dialogues sont maintenant découpés en « phrases ». Ces « phrases », prononcées par les personnages, sont associées à une ou plusieurs « questions », que peut poser le joueur. Chaque « question » agit comme un lien pouvant renvoyer à une autre « phrase », ou mettre fin au dialogue. Certaines « phrases » sont également associées à une « information », qui correspond au texte qui sera automatiquement noté dans le calepin du joueur. Enfin, certaines « questions » ne seront visibles que si une « phrase » particulière à déjà été rencontrée, ce qui permet de « débloquer » des « questions » au fur et à mesure de la partie. Nous utiliserons ce système pour inciter le joueur à réinterroger certains personnages. Nous avons précisé de la sorte l'ensemble des mécanismes de jeu pour les implémenter. Une fois qu'ils ont tous été intégrés à un prototype, nous avons évalué sa facilité de manipulation et son intérêt ludique. Nous avons alors corrigé le prototype jusqu'à obtenir des retours d'évaluations correspondant à nos attentes.

2.3.5 REDACTION DU SCENARIO

Une fois le principe de jeu validé par des prototypes, débute la rédaction du scénario. Si les premiers prototypes ne possédaient que trois lieux pour autant de personnages afin d'évaluer le principe de jeu, il faut maintenant intégrer la totalité du contenu. En accord avec les experts du domaine « sérieux », nous avons défini 23 lieux possédant chacun un personnage à interroger. Au total, 188 « phrases » de dialogue ont du être écrites pour ces 23 personnages. 61 de ces phrases sont associées à des « informations » que peut noter le joueur, ce qui signifie donc qu'*Eco-Reporter* a été conçu pour diffuser 61 « valeurs » définies dans le contenu sérieux. Ce jeu vise également à familiariser le joueur avec la géographie du quartier.

La rédaction de ces dialogues s'est tout d'abord appuyée sur un « prototype papier ». Nous avons créé une « carte » indexant, pour chaque personnage, ses liens avec les autres protagonistes et les « thèmes » du contenu sérieux qu'il devait aborder dans son dialogue. Cette carte matérialise également les « questions » qui doivent être débloquées durant le jeu. La création d'un tel document fut tout d'abord motivée par des fins de conception : nous avions besoin de nous repérer dans la structure non-linéaire du scénario. Mais il fut également utilisé tel quel pour exposer le scénario du jeu aux autres membres du projet. Cela nous renvoie au potentiel des outils théoriques de conception pour la communication entre les différents membres d'une équipe *(p.82)*. Notons d'ailleurs que des projets d'outils techniques dédiés au Serious Game commencent à intégrer de tels outils de réflexion *(p.222 et p.225)*.



42. *Eco-Reporter* : extrait de la « carte » synthétisant le scénario du jeu

Le cycle de prototypage s'est alors poursuivi pour évaluer la pertinence du scénario une fois intégré au jeu. Nous avons également intégré les éléments graphiques et sonores réalisés par des prestataires externes. La réalisation et l'évaluation de ces prototypes permirent d'aboutir à un Serious Game considéré comme « complet » selon la méthodologie de ***Fullerton*** *(p.61)*.

2.3.6 EXPLIQUER LE FONCTIONNEMENT DU JEU A TRAVERS LE JEU LUI-MEME

Un dernier aspect fut alors évalué par les cycles de prototypages : l'explication du fonctionnement du jeu. De très nombreux ajustements furent nécessaires pour expliquer « comment jouer » à un public qui n'avait jamais touché de jeux vidéo auparavant. L'approche consistant à s'appuyer sur un genre vidéoludique existant à certes des avantages. Mais quand le jeu se destine à un public novice en la matière, il est facile de se laisser piéger quant aux connaissances préalables des joueurs. Ainsi, les phases de dialogues avec les personnages dans *Eco-Reporter* se déroulent comme dans la plupart des jeux d'aventure : un personnage est affiché à l'écran accompagné d'une bulle dans laquelle sont affichées les « phrases » du personnage. Ensuite, sous ce personnage, se trouvent plusieurs « questions » sur lesquelles le joueur peut cliquer pour poursuivre le dialogue. Le fait de cliquer sur une « question » renvoie à une nouvelle « phrase » et affiche de nouvelles « questions ». Ce principe de fonctionnement fait partie des « codes » du genre des jeux d'aventure. Pourtant, plusieurs de nos testeurs n'ayant jamais joué à un jeu vidéo auparavant ont été mis à mal par ce principe. Ils n'avaient tout simplement pas remarqué que les « questions » changeaient avec chaque « phrase ». Ainsi, au lieu de dialoguer avec un personnage en sélectionnant à chaque fois la « question » la plus pertinente par rapport à la « phrase » courante du personnage, il se contentaient de cliquer tout d'abord sur la première « question », puis sur la deuxième, sur la troisième, et enfin sur la quatrième qui permet par convention de quitter le dialogue. Alors que la plupart des personnages sont dotés de plusieurs dizaines de « phrases » reliées entre elles selon une arborescence non-linéaire, ces testeurs ne voyaient que les trois mêmes « phrases » pour chacun des personnages. Ils passaient ainsi complètement à côté de l'intérêt du jeu. Nous avons remédié à ce problème en proposant que le premier personnage du jeu serve de « guide », et explique de manière interactive le fonctionnement du système de dialogue. De même, la première fois qu'un joueur arrive sur un des quatre « écrans » de jeu *(carte pour se déplacer, lieu pour prendre des photos, dialogue avec un personnage, et*

interface de rédaction), une aide contextuelle s'affiche en surimpression pour expliquer le fonctionnement des différents boutons.

43. *Eco-Reporter* : explication du principe des dialogues *(gauche)* et aide de jeu contextuelle *(droite)*

Loin d'être triviale, l'intégration de tels mécanismes destinés à expliquer le fonctionnement du jeu à un public n'ayant potentiellement jamais touché de jeux vidéo fut à l'origine de nombreux cycles itératifs de prototypage et test. Une première version du jeu expliquait successivement le fonctionnement de trois « écrans » dès le démarrage du jeu. Ces trois explications consécutives se sont avérées constituer trop d'informations à assimiler pour les testeurs. Nous avons donc réduit cette phase de présentation à deux écrans seulement, l'écran de la carte n'étant présenté au joueur qu'après un premier dialogue avec un personnage. Cela permet de « retenir » l'attention du joueur en le laissant jouer un peu avant de lui expliquer de nouveaux mécanismes de jeu. De même, un prototype affichait une aide contextuelle qui demandait au joueur de cliquer sur l'écran pour la faire disparaître avant de pouvoir continuer à jouer. Cela s'est avéré perturbant pour certains testeurs, pour qui ce « clic supplémentaire » était une source d'inquiétude : face à une action de leur part qui n'avait pas l'air de faire avancer le jeu, ils commençaient à douter d'avoir bien assimilé comment jouer et hésitaient à cliquer de nouveau, cherchant plutôt à réafficher l'écran d'aide.

Au final, que ce soit pour expliquer des mécanismes de jeu que l'on pourrait penser comme « évidents » car faisant partie de la culture vidéoludique, ou pour ajuster la pertinence et le débit des explications de jeu, l'intégration d'un « tutorial » au sein même du jeu s'est révélé être une tâche très importante du processus de création de ce Serious Game. De nombreux tests ont été nécessaires afin d'arriver à mettre en place des explications qui soient délivrées au bon moment pour qu'un joueur novice puisse appréhender le fonctionnement du jeu sans aide extérieure *(p.147)*.

2.4 LE JARDINIER ECOLO

En complément d'un jeu d'aventure simple d'accès et visant les joueurs attirés par l'aspect « réflexion », nous avions également envisagé de proposer plusieurs jeux d'envergure plus modestes mais mettant l'accent sur le côté « action ». La dimension exploration et les nombreux textes de dialogues seront ici remplacés par des manipulations à effectuer en temps limité dans des jeux nécessitant de développer une stratégie élaborée pour gagner des points de « score ». Si *Eco-Reporter* vise à transmettre de nombreuses informations sur la vie quotidienne dans l'écoquartier *Andromède*, ces autres jeux traiteront des différents aspects de la construction de ce quartier. Le choix d'une orientation « action » limite la quantité d'information transmissible avec chaque jeu, car nous souhaitons toujours nous inscrire dans une approche « intrinsèque » où le contenu sérieux est mélangé au système de jeu *(p.106)*. Concrètement, nous nous limiterons à un seul « message » par jeu. Mais ce message devra

être assimilé pour gagner, car il incarnera la stratégie gagnante du jeu. Par exemple, dans un jeu portant sur les espaces verts, le message sera le fait que les plantes exotiques consomment beaucoup d'eau et ne sont pas forcément pertinentes dans un écoquartier. Pour gagner à ce jeu, le joueur devra donc comprendre qu'il ne faut pas utiliser de plantes exotiques dans l'aménagement de son espace vert.

2.4.1 PLUSIEURS CONCEPTS DE JEUX

Selon ce principe, nous avons alors proposé de nombreux concepts de jeux, tous basés sur un des nombreux « métiers » impliqués dans la création de l'écoquartier *Andromède* :

- **Le jeu des espaces verts :** ce jeu propose au joueur d'aménager et d'entretenir un espace vert. Le joueur se verra proposer plusieurs espèces de plantes *(exotique, locale…)* qu'il pourra planter dans un espace vert vierge. Une fois plantées, ces plantes ont besoin d'eau pour rester vigoureuses. Régulièrement, des visiteurs traverseront le parc. Si les plantes sont bien arrosées, elles auront un effet positif sur le moral des visiteurs. Si elles sont en manque d'eau, elles auront un effet négatif sur le moral des visiteurs. Plus les visiteurs sortiront contents du parc, plus le joueur gagnera de points. Les espèces exotiques de plantes ont un effet plus important sur le moral des visiteurs, mais sont nettement plus gourmandes en eau. La stratégie gagnante du jeu consiste donc à comprendre qu'il vaut mieux planter des espèces locales que des plantes exotiques. Le joueur peut également « améliorer » ses plantes, ce qui a pour effet de les faire grandir. Or, un arbre étant moins gourmand en eau une fois qu'il a grandi, la stratégie gagnante consistera également à planter et faire grandir seulement quelques plantes, plutôt que planter de très nombreuses jeunes pousses qui seront très gourmandes en eau. Ce jeu s'inspire plus ou moins du genre vidéoludique *« Tower Defense »*, notamment de *Flash Element TD* ***(David Scott, 2007)***

- **Le jeu des chantiers :** A partir d'un terrain vierge, le joueur doit construire des maisons pour les habitants. Il peut pour cela diriger deux équipes d'ouvriers : les constructeurs et les nettoyeurs. Les constructeurs ont la capacité de générer des maisons, mais ce faisant il créent des déchets et produisent du bruit. Les nettoyeurs peuvent enlever ces déchets et poser/enlever des panneaux anti-bruit. Des habitants arrivent de manière régulière. Le joueur doit leur construire des maisons jusqu'à ce que la zone de jeu soit totalement remplie. Dès qu'une maison est construite, des habitants l'occupent. Si un chantier se déroule à côté d'une maison habitée, ses occupants seront incommodés par le bruit et les déchets. S'ils sont trop incommodés, ils finiront par partir, laissant une maison vide et non réutilisable dans l'espace de jeu. Le joueur gagnera des points en fonction du nombre de maisons toujours habitées une fois que toute la zone est construite. Pour gagner, le joueur devra donc utiliser son équipe de nettoyage de manière à limiter les nuisances produites par l'équipe de construction, mais aussi planifier ses constructions de manière à gêner le moins d'habitants possibles. Ce jeu illustre la politique de « chantiers propres » mise en place sur *Andromède* afin de ne pas incommoder les premiers habitants alors que les constructions continuent. Ce jeu s'inspire du genre vidéoludique *« Time Management »*, et plus particulièrement de *Diner Dash* ***(GameLab, 2003)***.

- **Le jeu de l'urbaniste :** à l'échelle du quartier, le joueur sera chargé de disposer des blocs « maisons » et des blocs « routes ». Il est donc libre de disposer les différents « programmes de construction » dans le quartier. Mais ces programmes ont des formes géométriques différentes et leur placement demande donc réflexion, à l'image de *Tetris* ***(Alexey Pajitnov & Vadim Gerasimov, 1985)***. Une fois placé, un bloc commence à se dégrader. Il faut donc régulièrement remplacer les blocs posés par des blocs plus récents. Si le joueur place ses blocs de manière anarchique, il sera obligé de

détruire des programmes encore en bon état pour remplacer les blocs détériorés. Mais s'il adopte la stratégie de planification urbaine dite en « îlots », qui fut utilisée pour *Andromède*, il pourra plus facilement entretenir les bâtiments de son quartier. Ce jeu illustre donc la méthode d'urbanisation modulaire d'un écoquartier, qui tranche avec les lotissements résidentiels des années 90, afin de pouvoir plus facilement rénover les infrastructures dans le temps. Ce jeu s'inspire librement du Serious Game *City Rain* ***(Mother Gaia Studio, 2009)*** et du jeu vidéo *Sim City* ***(Maxis, 1989)***.

- **Le jeu de l'architecte :** un petit jeu très simple destiné à illustrer les avantages des écomatériaux dans la construction. Le joueur incarne l'architecte d'un immeuble, et se voit doté d'un budget limité. Il peut alors choisir les matériaux à utiliser pour isoler l'immeuble, dans une liste allant des matériaux de base *(peu onéreux)* à des matériaux écoperformants *(mais cher)*. Une fois les matériaux choisis, le joueur dispose d'un curseur permettant de régler le niveau de chauffage et de climatisation de l'immeuble. Un thermomètre indique en temps réel la température de l'habitation. Les jours et les saisons défilent alors que le joueur doit essayer de maintenir une température intérieure agréable pour les habitants. S'il a choisi des matériaux d'isolation basiques, il aura certes réalisé des économies au départ mais réalisera bien vite que sa facture de chauffage et de climatisation sera très élevée. A l'inverse, en choisissant d'investir dans des écomatériaux, le joueur réalisera des économies de chauffage et de climatisation qui se révéleront plus intéressantes d'un point de vue économique au fur et à mesure que la partie se déroule.

- **Le jeu de l'aménageur :** ce jeu vise à illustrer la manière dont la ***SEM Constellation***, partenaire du projet et aménageur du quartier *Andromède*, a sélectionné les différents programmes qui y sont installés. Le joueur est face à une carte de l'écoquartier *Andromède*, avec tous ses lots qui sont libres. Il reçoit alors des propositions de programmes *(bureaux, habitations, infrastructures publiques, espaces verts…)* qui sont définies par plusieurs caractéristiques : *coût, performances écologiques…* Le joueur est libre de refuser ou d'accepter ces offres, auquel cas il devra placer ledit programme dans un des lots du quartier. Comme dans la réalité, le jeu se déroule en trois « tranches » : le joueur place des programmes sur un tiers de la superficie du quartier, puis reçoit un bilan avant d'aménager le tiers suivant. Selon les choix du joueur, différents problèmes peuvent survenir : *inondations si le goudronnage des sols est trop important, plaintes des habitants s'il n'y a pas assez de commodités…* Ces problèmes seront programmés de manière à mettre en valeur les choix d'aménagement opérés pour l'écoquartier *Andromède*. Ce concept de jeu s'inspire très librement de *Sim City* ***(Maxis, 1989)***.

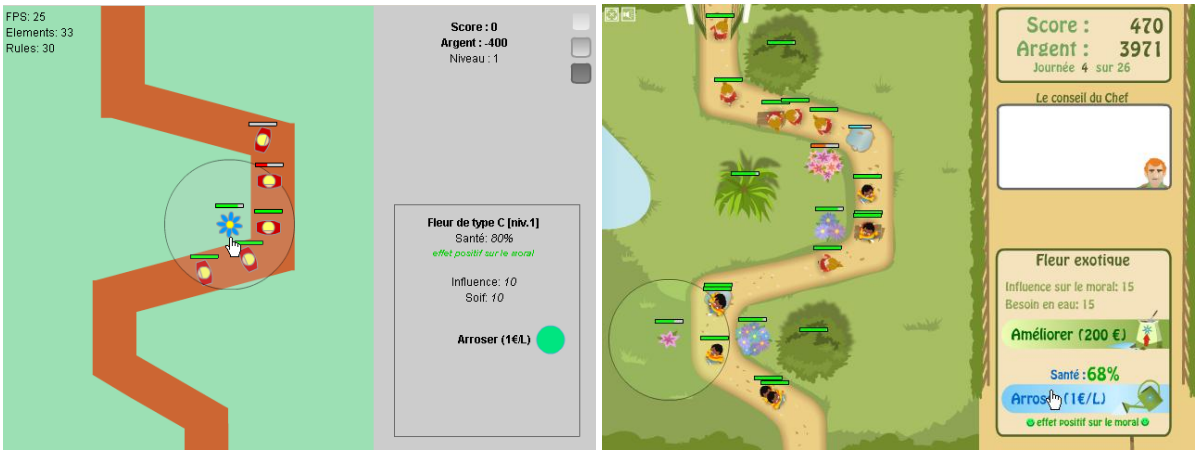
2.4.2 DES CONTRAINTES ECONOMIQUES LIMITANT L'AMPLEUR DU PROJET

Ces différents concepts de jeu ont été validés par les différents partenaires du projet. Cependant, bien que cela ne transparaisse pas dans les modèles théoriques de conception de Serious Games *(p.91)*, un tel projet est généralement soumis à des contraintes de temps et de budget. Ainsi, bien que ces cinq Serious Games paraissaient pertinents, la réalité économique du projet ne permettait pas de tous les développer. Nous avons donc été contraints de choisir un seul concept de jeu au sein de cette liste, afin de pouvoir réaliser un Serious Game « d'action » venant compléter l'approche de *Eco-Reporter*. Notre choix s'est porté sur le jeu des espaces verts, dont la création a été poursuivie sous le titre *Le Jardinier Ecolo*.

2.4.3 UNE « STRATEGIE DE JEU GAGNANTE » AU SERVICE D'UN MESSAGE SERIEUX

Nous avons utilisé un processus itératif, basé sur le *modèle générique DICE*, pour développer ce jeu grâce à de nombreux prototypes régulièrement évalués. Le fait de s'appuyer sur un genre de jeu existant, bien que ses codes furent quelque peu transformés pour l'occasion,

permettait de bénéficier d'une certaine base quant à la pertinence ludique du principe de jeu *(p.134)*. Nous avons donc pu nous concentrer sur l'affinage des mécanismes de jeu *(définition des différentes types de plantes, de leur coût, ajustement des vagues de visiteurs…)* afin de mettre en avant le message du jeu : l'utilisation d'espèces locales est écologiquement préférable pour l'aménagement d'espaces verts car elle permet d'économiser l'eau. De nombreux tests auprès de personnes variés en âge et en expérience dans la pratique du jeu vidéo ont été réalisés de manière à s'assurer que les joueurs développaient bien cette stratégie pour marquer un maximum de points. Si le principe de base du jeu fut rapide à poser, son « équilibrage » *(ajustement des coûts, de l'influence des plantes…)* pour pousser les joueurs vers cette stratégie sans pour autant les empêcher d'en tester d'autres fut particulièrement long : environ quatre mois, soit plus de la moitié du temps de création de ce Serious Game. Afin de mener les nombreuses corrections de prototypes inhérentes à ce mode de développement itératif, nous avons commencé à réaliser un outil technique destiné à faciliter la modification des règles de jeu *(p.256)*. Les différents cycles d'itération ont ainsi permis d'aboutir à un jeu « terminé », prêt à être diffusé auprès du grand public.



44. *Le Jardinier Ecolo* : prototype *(gauche)* et version finale *(droite)*

2.4.4 SUIVRE LES JOUEURS POUR EVALUER L'IMPACT DU MESSAGE SERIEUX

En ce qui concerne l'évaluation de la transmission du « contenu sérieux » du jeu, elle est ici très simple : étant donné qu'il faut obligatoirement comprendre le message du jeu pour développer une stratégie permettant de gagner beaucoup de points, le « score » des joueurs permet d'évaluer directement leur compréhension du message diffusé par ce Serious Game. Afin de pouvoir évaluer les joueurs, nous avons donc enregistré à intervalles réguliers les quelques variables permettant d'analyser leur stratégie de jeu : *score total, argent restant, nombre de plantes achetées, somme totale investie dans l'arrosage…* Concrètement, le jeu se compose de 26 niveaux, entrecoupés de trois « rapports d'activité » qui présentent ces variables au joueur, afin de l'aider à améliorer sa stratégie. Lorsqu'un rapport est présenté au joueur, les données qu'il contient sont également enregistrées par notre système d'analyse. A partir de l'analyse de ces données, nous pouvons alors évaluer simplement la transmission du contenu sérieux opérée par ce Serious Game *(p.139)*.

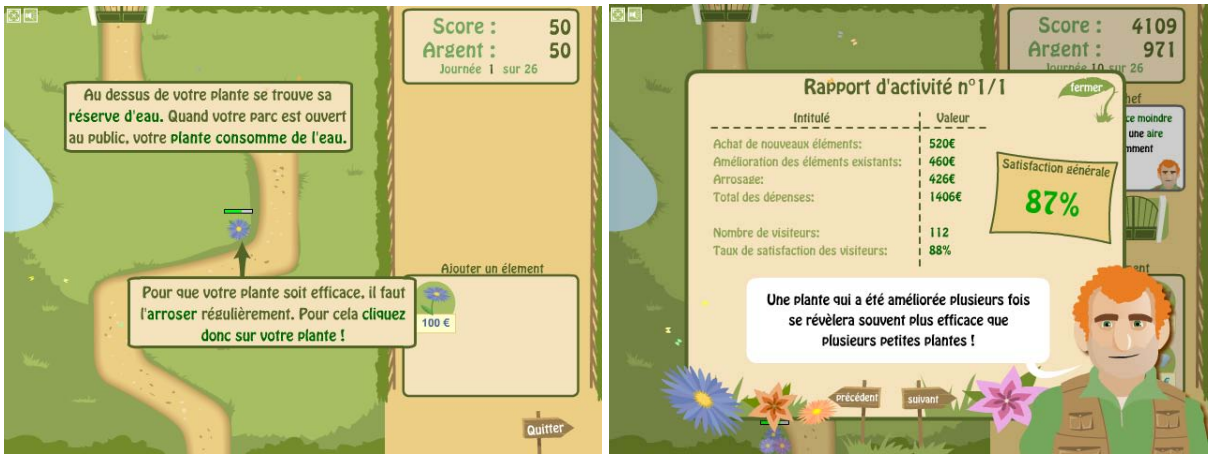
2.4.5 RECHERCHE D'UN MODE D'EXPLICATION ADAPTE AU JEU

Enfin, comme pour *Eco-Reporter*, les derniers cycles itératifs de prototypage du *Jardinier Ecolo* ont été consacrés à la mise en place d'un système d'explication du fonctionnement du jeu. Si *Eco-Reporter* était un jeu relativement simple en terme de mécanismes, *Le Jardinier Ecolo* s'appuie quant à lui sur un système plus complexe. Les nombreux éléments achetables possèdent des caractéristiques influant sur le moral des visiteurs et sur la consommation d'eau, un système de zone d'influence impacte le placement de ces éléments, plusieurs types de publics existent avec des réactions différentes… Pourtant, ce Serious Game doit rester accessible à des personnes n'ayant jamais utilisé de jeu vidéo auparavant.

Notre première approche fut de proposer un bouton « aide » sur l'écran d'accueil du jeu. Ce bouton menait à quatre écrans expliquant les différents mécanismes du jeu : *achat et placement de plantes, principe d'influence des plantes sur les visiteurs, moyens d'entretiens des plantes, possibilité d'améliorer les plantes pour modifier leur caractéristiques…* Nos tests ont clairement montré que la majorité des gens ne cliquaient pas sur ce bouton d'aide, même pour les personnes qui n'ont jamais joué à un jeu vidéo de leur vie. Le fait de rendre obligatoire la consultation de cet écran avant la partie fut loin de résoudre ce problème. Si les joueurs lisaient bien les explications et semblaient les comprendre, une fois arrivés dans le jeu nous constatons qu'ils étaient loin d'avoir tout retenu. Même en simplifiant les textes et en les illustrant par des exemples animés, la quantité d'information à transmettre sur le fonctionnement du jeu était tout simplement trop importante pour la plupart des joueurs.

A la recherche d'inspiration, nous avons alors analysé le jeu *Plant vs. Zombies* ***(PopCap Games, 2009)***, un titre du genre *« Tower Defense »* plébiscité pour son accessibilité. Nous avons étudié la manière dont ce jeu, à la complexité comparable à notre Serious Game, expliquait son fonctionnement à des joueurs novices. La solution développée par ce titre est le recours à un « tutorial » : un système de guidage intégré au jeu qui explique de manière interactive le fonctionnement du jeu durant les premières étapes de la partie. Si de nombreux jeux vidéo ont recours à ce principe pour expliquer « comment jouer » à tous les joueurs qui n'aiment pas lire de manuel, ce titre le fait de manière très progressive. Ainsi, les premiers niveaux de jeu se résument à effectuer des actions très simples : *planter une seule plante, récolter sa première ressource…* Les autres espèces végétales sont alors progressivement introduites au fur et à mesure que le joueur passe les niveaux. Cette approche possède deux avantages. D'un côté, elle permet d'expliquer « pas à pas » le fonctionnement d'un système complexe. Mais surtout, elle permet de proposer au joueur des niveaux de plus en plus difficiles, maintenant ainsi son intérêt pour le jeu en augmentant progressivement la complexité des défis qu'il doit relever *(notion de « flow » p.108)*.

Nous avons donc abandonné le « manuel » de l'écran d'accueil en faveur d'un tutorial intégré au *Jardinier Ecolo*. Afin de ne pas ennuyer les joueurs déjà familier avec le jeu, le jeu demande au joueur s'il souhaite une explication du fonctionnement du jeu. Si oui, des bulles sont incrustées à l'écran pour guider le joueur dans ses premières actions. Afin d'éviter de surcharger un novice par trop d'informations, nous masquons temporairement les boutons sur lequel le joueur ne doit pas cliquer. Si le joueur effectue l'action demandée par la bulle d'explication, une autre bulle d'explication apparaît, et ainsi de suite jusqu'à ce que toutes les règles de jeu lui soient expliquées. A la fin des explications, le joueur n'est pas contraint de commencer une nouvelle partie : il est déjà en train de jouer depuis plusieurs minutes. Ce tutorial l'a simplement guidé durant les premiers niveaux du jeu. S'il recommence une partie sans ce tutorial, il aura affaire aux mêmes niveaux de jeu, les explications interactives en moins. D'après nos tests, la pertinence d'une telle approche est manifeste. Aucun testeur ne s'est alors plaint de ne pas comprendre comment jouer, même si plusieurs cycles itératifs de prototypages ont bien évidemment été nécessaire pour arriver à un tutorial tout à fait pertinent dans la quantité et la temporalité des explications données. Nous avons également renforcé cette dimension « aide de jeu » dans les « rapports d'activité » qui sont présentés au joueur durant la partie *(p.126)*. En plus des différentes statistiques sur les sommes d'argent investies dans l'arrosage ou le taux de satisfaction des visiteurs, ces rapports affichent des conseils expliquant indirectement les mécanismes du jeu, tel que : *« une fois améliorés, les arbres consomment très peu d'eau »* ou *« une plante améliorée plusieurs fois sera plus efficace que plusieurs petites plantes »*.



45. *Le Jardinier Ecolo* : tutorial intégré au jeu *(gauche)* et rapport d'activité *(droite)*

Au final, l'efficacité de l'approche « tutorial » nous a été confirmée lors des tests qualitatifs menés sur ce Serious Game, où nous avons pu observer que des personnes n'ayant jamais utilisé de jeu vidéo de leur vie sont rentrées dans une attitude compétitive : accrochées au jeu, elles y ont rejouées plusieurs fois afin de *« réaliser le meilleur score »* *(p.132)*. Cela n'aurait pas été possible si ces personnes avaient eu du mal à saisir le fonctionnement du jeu lors de leur première partie. Comme pour *Eco-Reporter*, la création du *Jardinier Ecolo* illustre donc le rôle primordial de l'explication du fonctionnement d'un jeu vidéo lorsque l'on souhaite s'adresser à un public pas forcément familier avec le jeu vidéo *(p.147)*. Cela nous renvoie également au fait que pour gagner à un jeu vidéo, qu'il soit « sérieux » ou non, il faut tout d'abord apprendre à y jouer *(p.111)*.

2.5 DIFFUSION DES SERIOUS GAMES

Une fois la réalisation de ces deux Serious Games terminée, ils ont été diffusés par l'intermédiaire de la ***SEM Constellation*** et de la ***Mairie de Blagnac***. Rappelons que la cible première de ces jeux est le public local : l'objectif est de faire découvrir l'écoquartier *Andromède* et ses avantages aux habitants de la région toulousaine. Ainsi deux modes de diffusion ont été employés :

- *Par cédérom :* 150 cédéroms contenant les deux Serious Games ont été distribués gratuitement pour diffuser le jeu de manière locale, en particulier lors d'un évènement ayant eu lieu au centre culturel ***Odyssud*** : le festival *Luluberlu*. Il proposa de nombreuses animations à destination du jeune public du 28 au 30 mai 2010.
- *Par Internet :* les deux jeux sont également accessibles sur le site Internet de la ***SEM Constellation***. Cela permet tout d'abord de diffuser plus facilement le jeu à l'échelle locale *(par exemple dans les écoles de la ville de **Blagnac**…)*, puis de montrer le jeu à l'échelle nationale.

Le choix de la technologie *Flash* *(p.186)* permit d'envisager les deux modes de diffusion sans avoir à opérer de modifications techniques particulières aux jeux. Les responsables communication de la ***SEM Constellation*** se sont ensuite chargées d'annoncer la sortie de ces Serious Games aux médias locaux *(France 3 Sud, la Dépêche du Midi, 20 minutes…)*. En rencontrant les journalistes pour leur parler de ces jeux, nous avons d'ailleurs pu constater que le phénomène du *« Serious Game »* est loin d'être connu par tout le monde. Ces exercices de communication se sont donc accompagnés d'un résumé rapide des définitions et enjeux de la vague actuelle des Serious Games *(p.18)*.

2.6 RETOURS QUALITATIFS SUR CES DEUX SERIOUS GAMES

Pour étudier la réception de ces deux Serious Games et leur pertinence pour la transmission des messages définis, nous avons tout d'abord mené des entretiens qualitatifs.

2.6.1 METHODOLOGIE

L'étude qualitative de la réception de ces deux Serious Games a eu lieu selon des modalités identiques aux tests utilisateurs effectués durant les cycles itératifs de réalisation des jeux. Nous avons tout simplement observé une personne en train de jouer, en nous abstenant de tout commentaire. Nous avons demandé au joueur de penser à voix haute, afin de pouvoir noter ses différentes réflexions lors de l'utilisation du jeu. Une fois la partie terminée, nous lui avons posé la question *« Pensez-vous que ce jeu diffuse un message ? Si oui, quel est-il ? »* et noté sa réponse pour compléter le compte-rendu d'observation. Précisons que les tests effectués durant les derniers cycles itératifs du processus de réalisation des jeux ont également servi de base pour l'analyse du retour qualitatif. En effet, ces tests ayant eu lieu sur des prototypes quasi-finalisés, les observations effectuées s'avèrent aussi pertinentes que celles réalisées sur la version finale des jeux, en tout cas pour les aspects qui nous intéressent ici.

Au total, nous avons observé plus d'une trentaine d'utilisateurs aux profils variés. Notre panel comprenait par exemple un enfant de 5 ans ne sachant pas lire, un homme de 74 ans n'ayant jamais joué à un jeu vidéo ou encore une joueuse occasionnelle âgée de 28 ans. En constituant un tel panel, nous avons essayé de rassembler un échantillon représentatif du public ciblé par ce projet. Le tableau ci-dessous détaille le profil des différents testeurs utilisés pour les études de réception de ces deux Serious Games. Les âges indiqués sont approximatifs.

Tableau 5. Profils des utilisateurs observés durant les entretiens qualitatifs pour les Serious Games d'Andromède

Sexe	Age	Profession	Pratique du jeu vidéo	Jeu testé
Femme	59	Directrice de ludothèque	Grande connaissance du jeu « traditionnel », mais ne pratique pas le jeu vidéo.	Reporter Jardinier
Femme	24	Chargée de communication	Joueuse occasionnelle de petits jeux vidéo gratuits sur Internet ou téléphone portable	Reporter Jardinier
Femme	40	Chargée de communication	Ne joue pas aux jeux vidéo, mais son fils oui.	Reporter Jardinier
Homme	40	Game Designer	Joueur passionné et grande culture vidéoludique	Reporter
Homme	45	Professeur des universités	Joueur occasionnel de jeux vidéo	Reporter
Femme	26	Responsable de service culturel	Ne joue jamais aux jeux vidéo et ne s'y intéresse pas	Reporter
Homme	35	Paysagiste	Ne joue pas aux jeux vidéo, mais y jouait étant adolescent	Reporter Jardinier
Homme	40	Chef de projet	Ne joue pas aux jeux vidéo, mais travaille dans le domaine de la simulation professionnelle	Reporter
Homme	40	Commercial	Ne joue pas aux jeux vidéo, mais son fils oui.	Jardinier
Homme	40	Chef de projet	Joue de manière très occasionnelle	Reporter
Femme	28	Ludothécaire	Joueuse occasionnelle de jeux vidéo sur console de salon	Reporter Jardinier
Femme	24	Educatrice jeunes enfants	Joueuse occasionnelle de jeux vidéo sur ordinateur.	Reporter Jardinier
Femme	35	Hôtesse d'accueil	Ne joue pas aux jeux vidéo, mais son fils oui.	Reporter Jardinier

Femme	23	Ludothécaire	Joueuse occasionnelle de jeux vidéo sur console de salon	Reporter Jardinier
Femme	42	Ludothécaire	Ne joue pas aux jeux vidéo, mais ses enfants oui.	Reporter Jardinier
Femme	50	Médiathécaire	Joueuse occasionnelle de jeux vidéo sur ordinateur	Reporter
Femme	50	Puéricultrice	Ne joue pas aux jeux vidéo.	Reporter Jardinier
Homme	74	Retraité	Ne joue pas aux jeux vidéo	Reporter
Homme	35	Ingénieur	Joueur occasionnel de jeux vidéo sur ordinateur	Reporter
Homme	26	Ingénieur	Passionné de jeux vidéo, joue quotidiennement sur console de salon et ordinateur	Jardinier
Femme	27	Infographiste	Joueuse passionnée et grande culture vidéoludique	Jardinier
Homme	45	Professeur d'université Responsable politique	Ne joue pas aux jeux vidéo	Reporter Jardinier
Homme	55	Responsable politique	Ne joue pas aux jeux vidéo	Jardinier
Femme	27	Etudiante	Joueuse passionnée de jeux vidéo	Reporter Jardinier
Femme	55	Secrétaire	Ne joue pas aux jeux vidéo, mais son fils oui.	Reporter Jardinier
Femme	50	Aide-ménagère	Ne joue pas aux jeux vidéo	Reporter
Femme	72	Retraitée	Ne joue pas aux jeux vidéo	Jardinier
Femme	25	Juriste	Ne joue pas aux jeux vidéo	Reporter Jardinier
Garçons & Filles	5-12	Enfants observés lors du festival *Luluberlu*	Joueurs réguliers de jeux vidéo	Jardinier

Ayant largement profité de notre entourage professionnel pour réaliser ces évaluations, la part de femmes est plus élevée que celles des hommes parmi nos testeurs. Si les âges et professions sont plutôt variés, la plupart des gens que nous avons observés en train d'utiliser les Serious Games sur *Andromède* ne sont pas des joueurs passionnés de jeu vidéo. Certains d'entre eux n'ont même jamais joué à un jeu vidéo de leur vie, alors que pour d'autres cette pratique se limite à des parties de jeux sur *Wii* entre amis ou en famille. Nos Serious Games étant destinés aux personnes peu familières avec le jeu vidéo aussi bien qu'à ceux qui le pratiquent de manière occasionnelle, notre échantillon de testeurs « qualitatifs » représente finalement assez bien le public visé.

2.6.2 RETOURS QUALITATIFS SUR ECO-REPORTER

Pour *Eco-Reporter*, trois types de comportements ont été observés chez nos testeurs. Tout d'abord, certains joueurs n'ont pas du tout accroché au concept du jeu. 2 de nos 22 testeurs ont essayé le titre quelques minutes, ont trouvé sa manipulation aisée et son principe sympathique, mais *« ça ne leur plaisait pas de lire toutes ces informations »*. Ils ont trouvé le jeu trop simple, pas assez *« ludique »* à leur goût. Un de ces testeurs était un joueur passionné de jeu vidéo qui n'aimait pas trop les jeux d'aventure car il les trouvait *« ennuyeux »*. L'autre testeur connaissait déjà bien l'écoquartier *Andromède*, et ne voyait donc pas d'intérêt particulier à relire des informations sur le sujet quand il pouvait aussi tester *Le Jardinier Ecolo*. Pour ces deux joueurs, la partie n'aura donc duré que quelques minutes.

Ensuite, des joueurs ont apprécié le côté « exploration de l'écoquartier » mais n'ont trouvé aucun intérêt à rédiger l'article. Ce comportement a été observé pour 4 de nos 22 testeurs. Ne rentrant pas dans l'optique de *« gagner le jeu »*, ces joueurs, tous des habitants de la ville de ***Blagnac***, ont été plutôt intéressés par *« la découverte de cet écoquartier en construction pas loin de chez eux »*. Ils ont appréciés la richesse des informations et ont pris de nombreuses photos et notes. Mais arrivé au moment de rédiger l'article, ils nous ont dit qu'ils trouvaient cette étape de composition légèrement *« prise de tête »* et s'estimaient déjà totalement satisfaits par les informations qu'ils avaient pu lire lors de la visite du quartier. A noter que ces testeurs ont donc exploré les 23 lieux du jeu et ont discuté avec tous les personnages. Pour ces joueurs, la partie aura durée entre 20 et 30 minutes.

Enfin, 16 de nos 22 testeurs ont littéralement été « accroché » par le jeu. Ils ont tout d'abord commencé par explorer le quartier, afin de récolter photos et informations. Cependant, nous avions incorporé un mécanisme qui rappelle aux joueurs, au bout de 12 minutes de jeu, qu'ils doivent composer un article et non pas seulement explorer le quartier. Concrètement, le jeu affiche une fenêtre avec un message du rédacteur en chef du journal pour lequel travaille le joueur, qui lui demande d'envoyer un brouillon du travail déjà accompli. La plupart des joueurs ont cliqué sur le bouton *« Je le ferai plus tard »*, en nous expliquant qu'ils souhaitaient tout d'abord finir de récolter des informations avant de rédiger. Mais ce message a clairement rappelé à nos testeurs quel était l'objectif du jeu. La plupart d'entre eux s'étaient laissé *« happer »* par la dimension exploration au point d'oublier qu'il faudrait ensuite rédiger un article, tel que nous l'avons constaté en les écoutant réfléchir à voix haute. Une fois l'exploration terminée, ces joueurs ont ouvert l'interface de rédaction d'article, et, à notre surprise, s'y sont presque plus amusés que durant la phase d'exploration. Peut-être est-ce du à la dominante féminine de notre corpus de testeurs, toujours est-il que ces derniers ont clairement exprimé qu'ils s'amusaient beaucoup à organiser les informations récoltées pour composer un article. Plusieurs méthodes ont été ici observées. Certains commençaient par disposer toutes les photos qu'ils souhaitaient utiliser, avant de rajouter des informations textuelles pour les *« garnir »*. D'autres ont simplement déposé les informations dans l'ordre par défaut proposé par le jeu, puis les ont illustrées avec des photos. Certains ont fait l'effort de relire toutes les notes avant de les agencer à leur convenance. De même, quelques utilisateurs n'ont utilisé les intertitres qu'après avoir placé des photos et les informations textuelles, alors que d'autres ont commencé par placer les intertitres afin de structurer leur article. Une fois cette étape terminée, tous ces testeurs avaient en général des articles qui pouvaient bénéficier d'une publication, mettant donc fin à la partie. Mais afin de contenter différents profils de joueurs *(ceux disposant de peu de temps et ceux pouvant jouer plus longtemps)*, nous avions mis en place trois « niveaux » de publication : un *« billet sur le site Internet du journal »*, un *« article dans l'édition papier régionale »*, ou le *« dossier du mois dans l'édition nationale »*. Pour accéder à chacun de ces niveaux de publication, le joueur doit créer un article répondant à des critères quantitatifs : si l'article contient X photos et Y informations textuelles, alors il peut bénéficier de tel niveau de publication. Nous n'avons volontairement mis aucun critère sur la nature des photos et informations composant l'article afin de laisser les joueurs choisir ceux qui leur plaisent le plus. Cela permet de se baser sur cette création d'article pour évaluer l'impact du jeu *(p.120)*. A nouveau, nous avons ici observé deux réactions. D'un côté, des joueurs qui ont accepté la publication au niveau *« Internet »*, car il n'avaient pas le temps de continuer à jouer. Tous ces joueurs ont été agréablement surpris de constater que le jeu proposait une version *« mise en page comme un vrai journal »* de leur article, certains l'ayant même imprimé. De l'autre côté, les joueurs disposant de plus de temps ou plus motivés par le principe du jeu ont fait le choix de continuer à jouer jusqu'à arriver à une publication de niveau *« national »*. Pour ces derniers joueurs, cela correspond à une durée de partie d'environ 1h30, un temps de jeu qu'aucun de ces joueurs n'aurait pensé consacrer au jeu, d'autant plus qu'ils *« n'ont pas vu le temps passer »*. Ils étaient donc vraisemblablement en situation de *« flow »* *(p.108)*. Tous ces

joueurs, captivés par la rédaction de l'article, ont parfois passé autant de temps à composer leur article qu'à explorer le quartier. Ils ont donc été largement exposés aux nombreux messages du jeu. En tout, les joueurs ayant réussi à faire publier un article ont passé entre 30 minutes et 1h30 à jouer à *EcoReporter*. Ce temps de jeu est plus que surprenant de la part d'un public n'étant pas forcément attiré par le jeu vidéo.

2.6.3 RETOURS QUALITATIFS SUR LE JARDINIER ECOLO

En ce qui concerne *Le Jardinier Ecolo*, les retours d'utilisateurs étaient sensiblement différents. Sur les 18 adultes qui ont testé ce jeu, une seule personne n'a pas du tout accroché, ne *« voyant pas l'intérêt du jeu »*. Une autre testeuse, joueuse passionnée de jeu vidéo, à quant à elle arrêté de jouer au bout de quelques niveaux, trouvant le jeu *« beaucoup trop simple avec pas assez de choses à gérer »*. Pourtant, ces deux personnes ont consacré plus de 1h30 à *Eco-Reporter* pour atteindre le niveau de publication *« national »*.

Les autres testeurs du *Jardinier Ecolo* ont tous été suffisamment captivés par le jeu pour le terminer, ce qui représente environ 30 minutes de jeu. Environ deux tiers de ces testeurs ont manifesté le souhait de recommencer une partie pour améliorer leur score. Lors de leur première partie, tous ces joueurs ont acheté des plantes exotiques, poussés par les conseils donnés en ce sens par le jeu. Comme ils possédaient également des plantes locales, qui sont moins efficaces pour remonter le moral des visiteurs mais nettement moins gourmandes en eau, ils ont alors pu constater à quel point les plantes exotiques coûtaient cher à entretenir. Des phrases de type *« je voudrais bien acheter un banc, mais je viens d'arroser et il ne me reste plus d'argent »* ou *« mais c'est pas possible, je viens d'arroser ma plante exotique et il faut déjà que je lui redonne de l'eau »* revenaient régulièrement lors de l'observation de ces testeurs. Lors de leur seconde partie, ils se sont donc généralement tourné vers les espèces de plantes « locales », et ont constatés qu'ils obtenaient un meilleur score à l'échelle d'une partie. A la fin de chaque partie, le jeu demande son nom au joueur afin de l'enregistrer dans un « tableau des meilleurs scores ». Ce système, grand classique des jeux d'arcade, fut également efficace pour ce Serious Game. Certains de nos testeurs se battaient littéralement pour arriver tout en haut de ce tableau de score. Ils ont pour cela continué à jouer une fois rentrés chez eux, en dehors des évaluations que nous avons mis en place, pouvant parfois jouer une quinzaine de parties afin d'atteindre un classement satisfaisant.

Au final, la plupart de nos testeurs ont réussi à percevoir le message du jeu. Comme nous l'a formulé une de nos testeuses très inspirée : *« Les arbres ça prend une place dans la vie des gens, des beaux espaces verts ça donne envie d'y aller, d'y rester et de renouer avec la nature quand c'est bien entretenu. Mais ça ne se fait pas tout seul, un parc c'est de l'entretien et c'est pas fait au hasard. Bref, la nature c'est joli mais c'est du boulot, et ce jeu ça fait prendre conscience que les espaces verts ne sont pas faits au hasard »*. Une autre joueuse fut beaucoup plus synthétique : *« les espaces verts sont bon pour le moral et les plantes exotiques sont une ruine »*. Bien qu'incorporé au sein des mécanismes de jeu, le message sur la valeur et le coût de l'eau pour l'entretien des plantes semble avoir été identifié par la plupart de nos testeurs ayant terminé le jeu : *« Le jeu dit qu'on peut pas faire ce qu'on veut avec la nature, il y a des contraintes de budget, l'eau ça coûte cher, il y a différentes plantes et tout ça »* ou encore *« On passe beaucoup de temps à arroser, j'ai eu l'impression de ne faire qu'arroser car ça revient souvent et c'est chiant, mais c'est vrai que du coup on voit bien le côté écolo »*. Le fait que ce message soit associé à une « stratégie gagnante » implique que les joueurs commencent par « perdre ». Ils entament ensuite un processus de réflexion qui les amène à découvrir une meilleure stratégie, et donc à prendre conscience du message en déconstruisant mentalement le fonctionnement du jeu. Bien qu'elle ne soit effectuée que par les joueurs qui sont suffisamment captivés par le jeu pour le terminer et avoir envie d'y rejouer pour améliorer leur score, cette approche semble plutôt pertinente pour diffuser des messages simples de manière « intrinsèque ». Cette *« rhétorique de l'échec »* est plutôt répandue au sein

des Serious Games (Maurin, 2010). Le fait de perdre pousse le joueur à réfléchir sur les raisons de son échec, et donc à analyser le fonctionnement du jeu, révélant ainsi le message « intrinsèque » qu'il vise à diffuser. ***Maurin*** identifie même des jeux sciemment conçus pour être impossible à gagner, à l'image de *September 12th* ***(Gonzalon Frasca, 2003)*** qui traite de la délicate thématique des réponses à apporter au terrorisme *(p.37)*. Dans un article intitulé *« Je perds, donc je pense »*, ***Lee*** (2003) explique que ces jeux impossibles à gagner vont sciemment à l'encontre du principe de « l'apprentissage par essais et erreurs », largement utilisé par les jeux vidéo de divertissement pour inciter le joueur à découvrir par lui-même la manière de gagner au jeu. Si nous avons déjà vu que cette dernière approche peut être bénéfique dans le cadre de l'éducation *(p.111)*, nous voyons ici qu'elle n'est pas la seule manière permettant d'inciter un joueur à réfléchir sur une thématique particulière.

Cependant, bien qu'ils aient tous identifié le message du jeu, plusieurs utilisateurs nous ont questionnés sur la nature « sérieuse » du jeu. Autant ils voyaient bien la qualité de « Serious Game » pour *Eco-Reporter*, autant *Le Jardinier Ecolo* leur est apparu comme un *« jeu normal »*. Aussi surprenant que cela puisse paraître, nous pouvons voir ici une des limites de l'approche « intrinsèque » : quand les dimensions « ludique » et « sérieuse » sont mélangées, le jeu ne sera pas forcément perçu comme « sérieux ». Ce qui peut nuire à la bonne réception du message. Nous avions pu observer un effet similaire lors de la présentation de *September 12th* à nos étudiants *(p.241)*. Alors que certains d'entre eux identifiaient bien la dimension critique et militante de ce titre, d'autres ont cru qu'il s'agissait d'un jeu incitant à la violence. Ce point nous renvoie à la question primordiale de la « réception » d'un message diffusé à travers un « média » tel que le jeu vidéo, thème d'étude particulièrement riche dans le champ disciplinaire des sciences de l'Information et de la Communication (Amato, 2008; Genvo, 2006). Par rapport à notre expérimentation, nous retiendrons que, malgré tous les efforts du concepteur d'un Serious Game, de nombreux paramètres liés à la posture, la culture, le contexte social… du joueur influent grandement sur la réception du « contenu sérieux » que le jeu cherche à véhiculer. Cela conforte donc le rôle primordial des itérations de l'étape *« Evaluer »* du processus de conception d'un Serious Game.

Enfin, lors du festival *Luluberlu* *(p.128)*, seul *Le Jardinier Ecolo* était présenté de manière à intéresser le public de cet évènement, majoritairement composé d'enfants. Les enfants observés durant ce festival étaient complètement captivés par l'aspect ludique du jeu, et s'y adonnaient donc avec plaisir malgré le côté « sérieux » affiché par notre stand. Lorsque les enfants étaient assez grands pour comprendre les subtilités du jeu *(à partir de 7-8 ans)*, ils restaient sur le stand jusqu'à la fin d'une partie *(soit environ 30 minutes de jeu en moyenne)*. Comme nous disposions de deux machines avec le jeu, il n'était pas rare de voir un groupe d'enfants venir pour se « défier » : chacun jouait une fois et essayait de faire un meilleur score que ses amis. Ce comportement compétitif à été observé aussi bien chez les garçons que les filles. La dimension esthétique du jeu joue également un rôle important auprès de ce public. Ainsi, un groupe de quatre garçons âgés de 8 à 9 ans reprenaient en cœur le son *« Yeah ! »* qui est émis par le jeu à chaque fois qu'un visiteur sort « content » du parc. De même, le jeu présente un type de public « dépressif », qui est plus difficile à satisfaire que les autres. Graphiquement, ces personnages sont habillés tout en noir, sont plutôt costaud, et marchent très lentement. Ces enfants ponctuaient l'arrivée de ces personnages par la phrase *« Ah tiens, voilà les gorilles ! »*. Ainsi, la jeunesse de ce public l'empêchait parfois de saisir toutes les subtilités du jeu. Dans un registre similaire, un enfant de 5 ans est venu sur le stand alors qu'il ne savait pas lire. Le « tutorial » expliquant comment jouer étant uniquement à base de texte *(p.126)*, cela représentait un obstacle évident. Qu'à cela ne tienne, sa grand-mère qui l'accompagnait lui lisait les textes à voix haute, et l'accompagnait durant les premiers niveaux du jeu. A une intervention près de notre part, cette aide fut suffisante pour que cet enfant s'amuse avec le jeu, même s'il était loin d'en saisir tous les principes. Ainsi, le simple fait d'arriver à planter des fleurs et de voir des personnages défiler lui procurait satisfaction, loin

des préoccupations du choix pertinent de fleurs pour réaliser le meilleur score. Un comportement diamétralement opposé fut observé chez un petit garçon de 10 ans, véritablement accroché au jeu. Sur une seule journée, il est revenu à trois reprises sur le stand afin de réaliser le meilleur score possible. Si, comme tous les autres joueurs du festival, il a complété les 26 niveaux du jeu lors de sa première partie, il adopta ensuite une démarche d'optimisation de son score. Si, arrivé au premier rapport de performance proposé par le jeu *(p.126)*, il constatait que son score était inférieur à celui de sa partie précédente, il quittait le jeu pour recommencer une nouvelle partie en adoptant une stratégie différente. Lors de sa seconde visite, il commença même à poser de nombreuses questions aux animateurs du stand pour savoir comment réaliser un meilleur score. Par l'intermédiaire de ce petit Serious Game, cet enfant de 10 ans fut ainsi amené à discuter des avantages et inconvénients des plantes exotiques par rapport aux espèces locales, ou encore de l'intérêt des arbres par rapport aux fleurs, avec les adultes qui animaient le stand du jeu.

2.7 RETOURS QUANTITATIFS SUR CES DEUX SERIOUS GAMES

Ces considérations qualitatives se complètent bien évidemment des données quantitatives récoltées à travers les mécanismes d'évaluation intégrés au jeu *(p.120 et p.126)*.

2.7.1 METHODOLOGIE

Face à l'éventualité d'une diffusion large de *Eco-Reporter* et *Le Jardinier Ecolo* sur plusieurs sites Internet, nous avons décidé d'expérimenter des outils spécialisés dans l'analyse du joueur *(p.147)*. Nous avons tout d'abord utilisé *Playtomic*[36] *(anciennement SWFStats)*, un service de *« suivi de joueurs »* destiné aux jeux vidéo de divertissement. Il permet d'enregistrer facilement toute valeur numérique souhaitée par le concepteur du jeu, et permet ensuite de l'analyser à travers différents tableaux et autres outils de visualisation graphique. Lors de la période de diffusion des Serious Games liés à *Andromède*, ce service novateur était en phase de *« bêta-test »*, ce qui ne fut pas sans conséquences. Lors de l'analyse des données, nous avons en effet pu nous apercevoir que les nombreuses mises à jour de ce service ont entraîné plusieurs pertes de données. De nombreuses parties n'ont tout simplement pas été enregistrées par ce service. Néanmoins, suffisamment de données ont pu être récoltées pour permettre l'analyse de nos joueurs. S'agissant du premier service de ce type, nous avons quand même décidé de l'utiliser pour effectuer nos mesures quantitatives.

Afin de ne pas nous appuyer uniquement sur un outil en cours d'élaboration, nous avons également utilisé le service *Google Analytics*[37]. Cet outil de mesure d'audience est normalement destiné à l'analyse de sites Internet. Nous avons du recourir à quelques artifices afin de l'utiliser pour mesurer des données extraites de nos deux Serious Games. Concrètement, *Google Analytics* permet d'enregistrer deux types de données. La première consiste à mesurer simplement le nombre de visites effectuées sur une « page » donnée. Nous avons donc du découper la structure de nos Serious Games en « pages » afin d'effectuer quelques mesures avec cet outil. Par exemple, chacun des lieux de *Eco-Reporter* a été considéré comme une « page », permettant ainsi de mesurer le nombre de fois que les joueurs se rendent dans chaque lieu. Il en a été fait de même pour les différents « niveaux » du *Jardinier Ecolo*. Ensuite, *Google Analytics* permet d'enregistrer des *« événements »*, qui associent un nom d'évènement à une valeur numérique, ouvrant ainsi la voie à des mesures similaires à celles proposées par *Playtomic*. Cependant, *Google Analytics* n'est pas destiné à suivre une application générant de nombreux évènements, mais des sites Internet engendrant seulement une poignée d'événements par visiteur. Ainsi cet outil est limité à l'envoi de 10 évènements à la seconde, ce qui complique l'envoi simultané de plusieurs variables. De plus,

[36] http://www.playtomic.com/
[37] http://www.google.com/analytics/

le nombre total d'évènements pouvant être envoyés durant une « session de navigation » est limité, signifiant que si la partie dure un peu trop longtemps, les événements ne seront plus envoyés. Nous avons pu constater les inconvénients d'une telle limite sur *Le Jardinier Ecolo*, les données enregistrées par *Google Analytics* s'arrêtant en général au deux tiers de la première partie. Par contre, cet outil propose également une analyse géographique de la ville d'origine des visiteurs particulièrement poussée, dont les données furent appréciés par les commanditaires pour estimer la portée de la « communication locale » de ce projet. Par contre, seul *Playtomic* semble à même de proposer une estimation de la durée des parties qui soit fiable.

Au final chacun des deux services que nous avons utilisé présente ses propres avantages et ses propres limites, mais ces dernières sont différentes. Nous avons donc utilisé simultanément *Google Analytics* et *Playtomic* pour enregistrer les valeurs numériques, et *Google Analytics* pour compter les visites dans les différents « lieux » définis pour ces deux Serious Games. La version Internet du jeu, ainsi que la version cédérom si une connexion Internet est disponible sur la machine, enregistrent de manière transparente les informations suivantes :

- Le contenu des articles rédigés sur *Eco-Reporter*, qui est enregistré lorsqu'un joueur soumet un article répondant aux critères lui permettant d'être publié *(p.130)*. Afin de pouvoir également analyser le comportement des joueurs qui ne rédigent pas d'article, le nombre de visites dans chaque lieu et de conversations avec chaque personnage sont enregistrés de manière continue durant la partie.

- Le score des joueurs, le pourcentage moyen de satisfaction des visiteurs, l'argent restant, l'argent dépensé pour acheter de nouveaux éléments, l'argent dépensé pour améliorer des éléments existants, l'argent dépensé en arrosage, le nombre de plantes et d'objets *(bancs…)* possédés par les joueurs du *Jardinier Ecolo*. Le jeu se compose de 26 niveaux, entrecoupés de trois « rapports d'activité » qui présentent ces données statistiques au joueur, assortis de conseils pour l'aider à améliorer sa stratégie *(p.126)*. Lorsqu'un rapport est présenté au joueur, les données qu'il contient sont également enregistrées par notre système d'analyse.

En utilisant conjointement ces deux systèmes de suivi des joueurs, nous avons enregistrés des données sur une période allant du *20 mai 2010* au *20 mai 2011*. Les retours quantitatifs sur les deux Serious Games d'*Andromède* ont donc été observés sur un an de diffusion.

2.7.2 RETOURS QUANTITATIFS SUR ECO-REPORTER

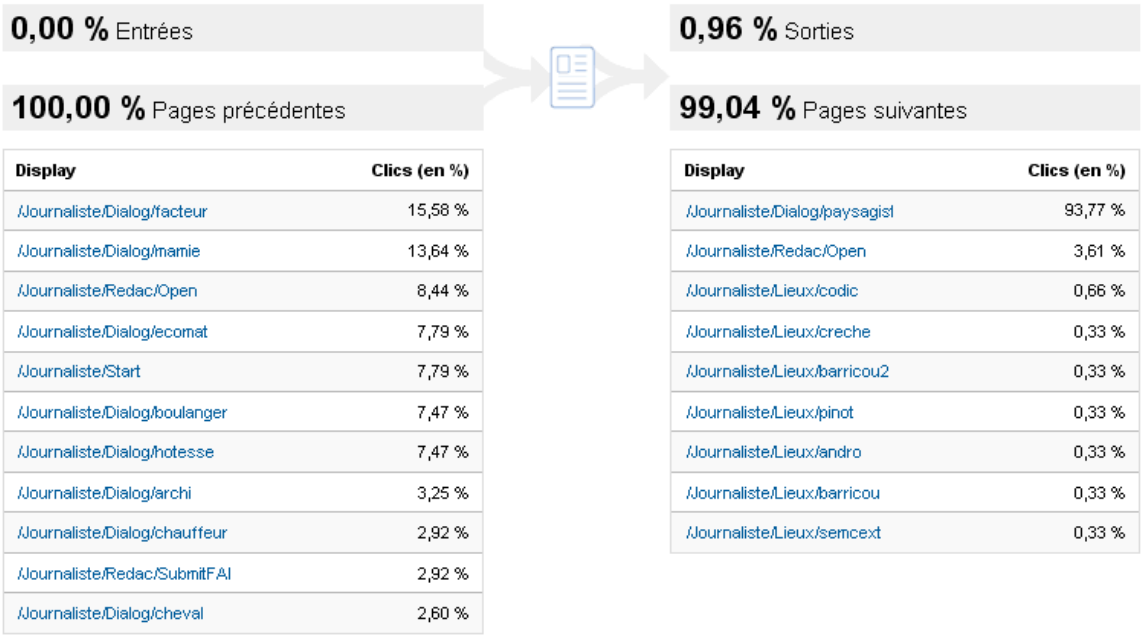
D'après *Google Analytics*, 1723 parties de *Eco-Reporter* ont été jouées sur la période d'observation, alors que *Playtomic* n'en a enregistré que 1082. A partir de ces données partielles, cet outil estime la durée moyenne d'une partie à 18 min 57.

Grâce à *Google Analytics* et son système de suivi des « pages », nous obtenons un aperçu des lieux visités par ces 1723 joueurs, et des personnages avec lesquels ils ont discuté. Comme nous pouvons l'observer dans le tableau ci-dessous, tous les personnages à l'exception du guide d'accueil ont été « visités » plus souvent que le lieu qui les abrite. Cela signifie tout simplement que les joueurs ont lancé plusieurs séances de discussion successives avec un même personnage. En effet, il arrive parfois qu'un dialogue se termine par manque de questions à poser. Les joueurs ont donc parfois relancé immédiatement un dialogue afin de poser des questions différentes, et ainsi récolter des informations supplémentaires. *« L'Espace Andromède »* est un cas à part car il s'agit du lieu de départ des joueurs, et que le personnage qui y est présent ne sert qu'à donner des indications sur la manière de jouer *(p.122)*. Comme nous l'avons observé durant l'étude qualitative, de nombreux joueurs reviennent plus tard sur ce lieu afin d'en prendre une photo, sans discuter à nouveau avec le personnage.

Tableau 6. Données quantitatives relatives aux visites des lieux de *Eco-Reporter*

Nom du lieu *(personnage associé)*	Visites du lieu	Dialogues avec le personnage
Espace Andromède *(Guide d'accueil)*	1878	1723
L'Ecrin Végétal *(Facteur)*	1071	1320
Cap Constellation *(Paysagiste)*	851	916
Avenue d'Andromède *(Boulanger)*	787	801
Complexe Sportif d'Andromède *(Entraîneur de football)*	773	842
Le Grand Patio *(Retraité vivant en appartement)*	759	860
Gendarmerie *(Gendarme)*	746	792
Cours Barricou *(Enfant jouant au football)*	732	845
Promologis *(Jeune couple d'étudiants)*	670	782
Centre de Maintenance du Tramway *(Conducteur de Tramway)*	624	646
Sirrah *(Institutrice)*	605	720
La coulée verte *(Brigadier de la police montée)*	556	589
Lycée St-Exupéry *(Lycéenne championne de patin à glace)*	535	617
Théâtre de verdure *(Retraitée vivant en maison de retraite)*	501	585
Centre de loisirs Barricou *(Animateur du centre de loisirs)*	478	542
Le Galilée *(Architecte)*	472	488
Osmose (*Constructeur d'écobâtiments)*	462	513
Les Jardins Andalous *(Supporter du Blagnac Football Club)*	424	428
Carré Mondrian *(Hôtesse de l'air)*	371	471
Les Essentielles *(Directrice de la ludothèque)*	369	387
Crèche Cassiopée *(Educatrice jeunes enfants)*	335	364
Cours Pinot *(Famille franco-allemande avec un enfant)*	330	379
Centre de Formation des Apprentis de l'Industrie *(Enseignant)*	327	349

Notons également que *Google Analytics* propose un outil de visualisation fort pratique permettant, pour chaque « page », de savoir quelles sont les pages d'origines des visiteurs et leurs pages de destination. Nous pouvons donc savoir, pour chacun des « lieux » et « personnages » d'*Eco-Reporter*, quel est le parcours suivi par le joueur pour y arriver. Dans l'exemple ci-dessous, nous voyons que 16% des joueurs sont allés visiter le lieu *« Cap Constellation »* directement après avoir discuté avec le personnage du *« Facteur »*. Une fois arrivés sur ce lieu, 94% des joueurs ont discuté avec le personnage qu'il abrite, 4% des joueurs ont ouvert l'interface de rédaction d'article, tandis que 1% des joueurs semblent avoir quittés le jeu à ce moment là. Ce type de visualisation s'avère fort pratique pour une analyse détaillée du parcours des joueurs, et permet de placer *Google Analytics* dans la liste des outils de suivi des joueurs *(p.145)*



Display	Clics (en %)
/Journaliste/Dialog/facteur	15,58 %
/Journaliste/Dialog/mamie	13,64 %
/Journaliste/Redac/Open	8,44 %
/Journaliste/Dialog/ecomat	7,79 %
/Journaliste/Start	7,79 %
/Journaliste/Dialog/boulanger	7,47 %
/Journaliste/Dialog/hotesse	7,47 %
/Journaliste/Dialog/archi	3,25 %
/Journaliste/Dialog/chauffeur	2,92 %
/Journaliste/Redac/SubmitFAI	2,92 %
/Journaliste/Dialog/cheval	2,60 %

Display	Clics (en %)
/Journaliste/Dialog/paysagist	93,77 %
/Journaliste/Redac/Open	3,61 %
/Journaliste/Lieux/codic	0,66 %
/Journaliste/Lieux/creche	0,33 %
/Journaliste/Lieux/barricou2	0,33 %
/Journaliste/Lieux/pinot	0,33 %
/Journaliste/Lieux/andro	0,33 %
/Journaliste/Lieux/barricou	0,33 %
/Journaliste/Lieux/semcext	0,33 %

46. Exemple de visualisation détaillée du parcours des joueurs grâce à *Google Analytics*

Toujours d'après *Google Analytics*, nous constatons que 339 parties d'*Eco-Reporter* ont abouties à la rédaction d'un article publié. Cela signifie qu'environ 20% des joueurs d'*Eco-Reporter* ont été suffisamment captivés par le jeu pour aller jusqu'à la publication d'un article. En effet, lors de l'étude qualitative nous avions observé deux types de comportements : des joueurs auquel le principe de jeu ne plaît pas, et d'autres qui sont littéralement « happés » par l'aventure qu'il propose *(p.130)*. Cela se vérifie en regardant les données enregistrées sur la durée d'une partie, avec d'un côté de nombreuses parties qui ne durent que quelques minutes, et de l'autre des parties à la durée comprise entre 30 minutes et 1h30. Au total, 1631 articles ont été rédigés et proposés par les joueurs pour publication. Si seuls 339 de ces articles ont réussi à passer les critères de publication, nous observons logiquement que les articles simples *(niveau « Internet »)* sont plus nombreux que ceux du niveau *« national »*, dont la création correspond à une durée de partie d'au moins 1h30. A noter également que 42% des joueurs ayant réussi à faire publier leur article ont décidé de l'imprimer sur papier pour garder un souvenir du jeu.

Tableau 7. Données quantitatives relatives à la rédaction d'un article dans *Eco-Reporter*

Etape de la rédaction d'un article	Nombre total d'actions	Taille moyenne d'un article
Ouverture de l'interface de rédaction d'article	4026	-
Soumission de l'article au rédacteur en chef	1631	-
Refus de l'article par le rédacteur en chef	1284	-
Acceptation de l'article par le rédacteur en chef	339	48 éléments
Articles acceptés au niveau « Internet »	179	27 éléments
Articles acceptés au niveau « Régional »	98	63 éléments
Articles acceptés au niveau « National »	61	84 éléments
Impression sur papier de l'article accepté	145	-

Les données enregistrées nous permettent également de visualiser les « thèmes » du contenu sérieux *(p.116)* les plus utilisés par les joueurs. Nous constatons ici l'intérêt de l'approche « intrinsèque », grâce au principe de rédaction d'article qui permet d'évaluer quels sont les « messages » du contenu sérieux que les joueurs vont utiliser *(p.120)*. Tout d'abord, visualisons la répartition globale des différentes thématiques du contenu sérieux dans les articles composés par les joueurs :

Tableau 8. Répartition des thématiques pour les articles rédigés dans *Eco-Reporter*

Thème des messages utilisés dans l'article	Nombre total de messages cités	Nombre de messages différents dans le thème
Qualité de vie	1445	8 messages
Commodités	1228	6 messages
Généralités	1179	7 messages
Les espaces verts	1110	6 messages
Transports	1016	5 messages
Aspects Environnementaux	941	6 messages
Vie locale	840	6 messages
La Géothermie Profonde	666	5 messages
La gestion des eaux	597	5 messages
Mixité sociale et générationnelle	503	4 messages
Les Toitures Végétalisées	314	3 messages
Photographies des lieux	4021	27 photos

Ensuite, pour chacun de ces « thèmes », il est possible d'obtenir des statistiques plus détaillées. Tout d'abord, sur le nombre de messages d'un thème donné qui sont présents dans chacun des articles finalisés par les joueurs :

Tableau 9. Détail du nombre d'utilisations des messages de la thématique « Aspects Environnementaux »

Nombre de messages liés au thème dans l'article	Articles publiés
1 message lié aux *« Aspect Environnementaux »* dans l'article	31
2 messages liés aux *« Aspect Environnementaux »* dans l'article	49
3 messages liés aux *« Aspect Environnementaux »* dans l'article	43
4 messages liés aux *« Aspect Environnementaux »* dans l'article	52
5 messages liés aux *« Aspect Environnementaux »* dans l'article	44
6 messages liés aux *« Aspect Environnementaux »* dans l'article	42

Et enfin, il est tout simplement possible d'obtenir, pour chacun des 61 « messages » définissant le contenu sérieux d'*Eco-Reporter*, le nombre de fois qu'il a été employé dans les articles des joueurs :

Tableau 10. Détail de l'utilisation de chaque message de la thématique « Aspects Environnementaux »

Message lié au « Aspects Environnementaux »	Citations totales
« De nombreux bâtiments de cet écoquartier répondent à la certification Bâtiment Basse Consommation (BBC) »	200
« La certification BBC est attribuée aux bâtiments à faible dépense énergétique en terme de chauffage et d'éclairage »	186
« La limitation du goudronnage des sols permet de mieux gérer les espaces verts et la récupération de l'eau de pluie »	169
« Les parkings des équipements publics sont mutualisés pour accueillir les visiteurs tout en limitant l'imperméabilisation des sols »	145
« Pour les habitants, la certification BBC représente des économies sur le chauffage et la climatisation »	126
« Un éco-bâtiment est construit avec plusieurs techniques telles que les toitures végétales, la géothermie ou les panneaux solaires »	115

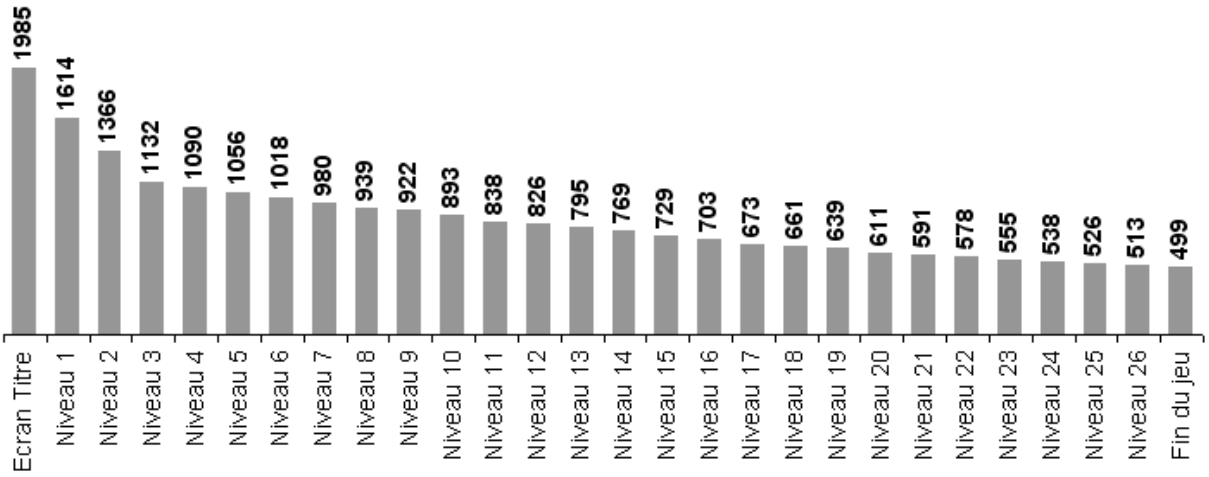
Grâce à l'intégration de méthodes simples de suivi des joueurs au sein même du principe du jeu, nous avons pu obtenir des données quantitatives intéressantes à deux niveaux. Tout d'abord, elles permettent au concepteur du jeu d'avoir une idée du parcours suivi par les joueurs. Il peut donc détecter les parties du jeu les moins populaires, afin d'éventuellement rééquilibrer le jeu lors de sa création. Une fois le jeu terminé et diffusé au grand public, ces mêmes méthodes de mesure permettent de fournir au commanditaire du projet un retour

quantitatif sur l'exposition des joueurs aux divers messages qu'il a souhaité diffuser à travers un Serious Game.

2.7.3 RETOURS QUANTITATIFS SUR LE JARDINIER ECOLO

D'après *Google Analytics*, 1985 parties du *Jardinier Ecolo* ont été jouées sur la période d'observation, alors que *Playtomic* n'en a enregistré que 1400. A partir de ces données partielles, cet outil estime la durée moyenne d'une partie à 18 min 10.

En nous appuyant sur les données récoltées par *Google Analytics*, nous pouvons avoir une idée du nombre de joueurs ayant terminé chacun des niveaux du jeu. Au total, si 1985 joueurs ont démarré le premier niveau du jeu, seulement 499 d'entre eux sont arrivés jusqu'à la fin. Cela signifie donc qu'environ 25% des joueurs qui commencent une partie la poursuivent jusqu'à la fin du jeu. Si le fait que trois quarts des joueurs quittent le jeu avant la fin de la partie pourrait paraître considérable au premier abord, cette donnée se voit nuancée par l'étude qualitative des joueurs *(p.132)*. En effet, nous avons observé que, dans un souci d'atteindre le meilleur score, certains joueurs n'hésitaient pas à recommencer une partie avant de l'avoir terminé, s'ils estiment que leur score dans les premiers niveaux du jeu n'est pas suffisant pour battre le record. A noter que les données enregistrées par *Playtomic* permettent d'obtenir une courbe similaire, malgré un nombre moindre de parties analysées. S'il est normal que le nombre de joueurs diminue légèrement de niveau en niveau (Cook, 2009), la diminution notable du nombre de joueurs sur les premiers niveaux aura ici nécessité une étude qualitative complémentaire pour être interprétée.



47. Nombre de joueurs ayant terminé chacun des niveaux du *Jardinier Ecolo*

Dues aux limitations propres à *Google Analytics* et son système « d'événements » *(p.134)*, seules les données enregistrées par *Playtomic* ont pu être utilisées pour étudier les stratégies développées par les joueurs à partir des différentes variables du jeu *(argent dépensé, taux de satisfaction des visiteurs…)*. Cependant, les données moyennes calculées automatiquement par *Playtomic* sont faussées car elles prennent en compte les jours où aucune partie n'a été jouée dans le calcul, rajoutant de nombreux « 0 » diminuant les valeurs moyennes. Nous avons donc du extraire manuellement les données brutes enregistrées de manière quotidienne par *Playtomic* et recalculer nous-même les valeurs moyennes pour chacune des variables enregistrées dans le tableau ci-dessous. Comme exposé précédemment, ces données sont enregistrées lors de « rapports d'activité » qui sont présentés aux joueurs à chaque tiers de sa partie *(p.134)*. Elles nous permettent de voir que plus la partie avance, plus les joueurs ont tendance à investir dans l'arrosage des plantes au détriment de leur achat. De même, les sommes investies par les joueurs dans l'amélioration des plantes existantes est quasiment équivalent au montant représenté par l'achat de nouvelles plantes. Ces deux données illustrent le fait que les joueurs semblent avoir plutôt bien assimilé le fait qu'avoir peu de plantes que

l'on fait grandir et que l'on entretient régulièrement est une meilleure stratégie que celle qui consiste à multiplier l'achat de plantes. De même, les équipements n'ont pas été négligés par les joueurs, qui ont du chercher à « rentabiliser » leurs quelques plantes par l'ajout de bancs permettant aux visiteurs de s'asseoir pour les admirer plus longtemps. Nous constatons aujourd'hui que ces données auraient été encore plus pertinentes si nous avions opéré une différenciation entre les espèces « exotiques » et « locales » en comptant le nombre de plantes achetées par les joueurs. Cette donnée n'étant pas pertinente pour les rapports d'activité *(un joueur pouvant visuellement voir à tout moment les différentes espèces de plantes sur son terrain)*, nous n'avions pas pensé à l'inclure avant la diffusion du jeu à grande échelle. Si elles ne permettent donc pas de rentrer autant dans le détail des stratégies des joueurs qu'une analyse qualitative *(p.132)*, ces quelques données quantitatives nous renseignent tout de même sur les tendances générales de leurs choix stratégiques.

Tableau 11. Données enregistrées sur *Le Jardinier Ecolo*

Donnée enregistrée	Rapport n°1	Rapport n°2	Rapport n°3	Moyenne
Somme dépensée pour l'achat d'éléments	636 €	1188 €	563 €	796 €
Nombre de plantes achetées	2,5	2,4	1,8	6,7
Nombre d'objets (bancs…) achetés	2,1	1,7	1,5	5,4
Somme dépensée pour l'amélioration d'éléments	421 €	957 €	554 €	644 €
Somme dépensée pour l'arrosage des plantes	510 €	1386€	1178 €	1025 €
Budget restant au joueur	176 €	231 €	211 €	147 €
Taux de satisfaction des visiteurs	83%	71%	59%	71%
Score du joueur	3454	13614	21899	21899

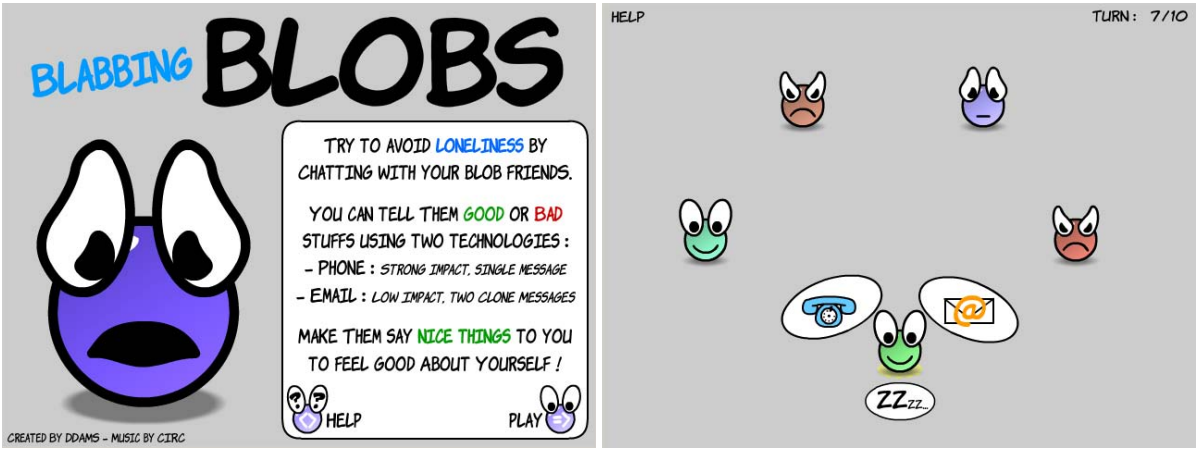
Au final, ces mesures quantitatives menées sur les deux Serious Games dédiés au quartier *Andromède* permettent d'enrichir les observations faites lors des évaluations qualitatives. Elles constituent la dernière phase de ce projet de 18 mois du contrat *CIFRE* dans lequel s'inscrit cette thèse. Ce projet nous ayant permis de suivre la création d'un Serious Game du début à la fin, il fut particulièrement riche en enseignements sur les spécificités de la conception d'un Serious Game. Impliquant des partenaires ainsi qu'une certaine obligation de résultats, nous avons néanmoins du faire certains compromis liés aux contraintes budgétaires et aux objectifs fixés en début de projet. Ce retour d'expérience s'accompagne donc de celui relatif à un autre Serious Game que nous avons réalisé de manière indépendante. Il nous a permis d'expérimenter une autre approche de conception : le « gameplay expérimental ».

2.8 CREATION DE SERIOUS GAME ET « GAMEPLAY EXPERIMENTAL »

Malgré les différences du processus de conception des deux Serious Games dédiés à l'écoquartier *Andromède* exposées dans ce chapitre, la création de ces deux jeux s'inscrit à chaque fois dans un genre vidéoludique existant. Nous avons d'ailleurs déjà évoqué les avantages *(p.119)* et les inconvénients *(p.122)* observés suite à une telle approche. Néanmoins, un concepteur de Serious Games n'est pas forcément obligé de s'inscrire dans un genre vidéoludique existant. Rien ne l'empêche d'inventer de nouveaux principes de jeu pour créer un Serious Game. Cette autre approche nécessite cependant de tester en profondeur la pertinence « ludique » de ce principe de jeu novateur. C'est pourquoi elle est généralement désignée par l'expression de *« gameplay expérimental »*.

Alors que l'approche « gameplay expérimental » n'apporte aucune garantie quant à la qualité « ludique » du jeu créé, le fait de s'appuyer sur un genre de jeu vidéo existant permet de bénéficier de l'expérience des autres jeux du genre. Cela permet également de s'appuyer explicitement sur la culture vidéoludique, ce qui peut représenter un avantage selon le public visé. Ces raisons expliquent sûrement pourquoi de nombreux Serious Games publicitaires, en

particulier ceux destinés au domaine alimentaire (J. Alvarez & Djaouti, 2010), s'appuient sur des genres de jeu populaires comme les jeux de plateforme ou de course. Il en va d'ailleurs de même pour les adaptations de films hollywoodiens en jeu vidéo (Blanchet, 2010). Dans une certaine mesure, nous pourrions même penser que le processus créatif visant à adapter un film en jeu vidéo est relativement proche de celui d'un Serious Game. L'intrigue et l'univers du film font alors office de « contenu sérieux » que le concepteur doit transmettre à travers un jeu vidéo. Si l'approche « gameplay expérimental » est donc visiblement loin d'être la plus répandue, elle reste cependant intéressante pour le Serious Game. Nous en avons personnellement fait l'expérience lors de la création d'un autre Serious Game : *Blabbing Blobs* ***(Damien Djaouti, 2008)***. Ce jeu a été créé pour un concours de création de Serious Games organisé par le site *WhoseGame* *(p.202)*. L'objectif du concours était de réaliser un jeu vidéo sur un des trois thèmes proposés (J. Alvarez & Maffiolo, 2011). Nous avons choisi de créer un jeu vidéo lié à la thématique *« lutter contre l'isolement par les technologies de l'information et de la communication »*. Mais au lieu de nous appuyer sur un genre vidéoludique existant, nous avons choisi d'expérimenter un principe de jeu innovant, dont les qualités ludiques resteraient à éprouver lors de sa création.



48. *Blabbing Blobs* : écran titre *(gauche)* et jeu *(droite)*

Ce Serious Game propose au joueur d'incarner un avatar parmi cinq « blobs communicants ». L'objectif du joueur est d'éviter que son avatar éprouve un sentiment de solitude à la fin de la partie. Une partie se déroule en dix « tours de jeu ». A chaque tour de jeu, chaque « blob » peut envoyer un email ou utiliser son téléphone pour communiquer avec un ou plusieurs autres blobs. Ce faisant, il peut envoyer des messages « positifs » ou « négatifs ». Le téléphone ne permet de contacter qu'un seul « blob », mais délivre un message dont l'impact émotionnel est plus fort que celui d'un email. En contrepartie, un email permet de contacter deux « blobs » à la fois. Les « blobs » joués par l'ordinateur ont une mémoire des messages qui leur sont envoyés, et auront donc tendance à développer un sentiment d'amitié ou d'inimité avec les autres « blobs ». La réception de messages positifs aura tendance à rendre un blob « heureux » alors que des messages négatifs auront tendances à le mettre en « colère ». Un blob heureux sera plus susceptible d'envoyer des messages positifs qu'un blob en colère. Dans tous les cas, le fait de recevoir un ou plusieurs messages de la part des autres blobs permet à chacun des blobs de lutter contre le sentiment de solitude qui ne cesse de les envahir. Ainsi, le joueur est amené à essayer de se « faire des amis » qui lui envoient des messages pour éviter de se sentir seul. Mais il sera en compétition avec les autres blobs qui cherchent également à se faire des amis. Il peut donc parfois être plus efficace d'énerver un blob pour s'en faire un « ennemi ». Certes, il enverra alors des messages « négatifs », mais cela permettra au joueur de diminuer le niveau de solitude éprouvée par son avatar. Ce Serious Game au principe expérimental nous a été inspiré par des sources diverses allant du réseau social *Facebook* aux relations humaines en général en passant par le jeu *Gossip* ***(Chris Crawford, 1983)***. Bien qu'expérimental au niveau des mécanismes de jeu, nous avons choisi

de nous inscrire dans une approche de conception « intrinsèque » *(p.106)*, intégrant donc notre message sur le rôle des technologies de la communication pour lutter contre la solitude au cœur du principe de jeu.

Cependant, et bien que nous ayons évalué le jeu en accord avec la méthodologie du *modèle générique DICE (p.103)*, ses mécanismes étaient loin d'être parfaits d'un point de vue ludique. Si le concept du jeu semblait présenter un potentiel certain, les testeurs de *Blabbing Blobs* ont clairement mentionné que le jeu n'était *« pas si amusant que cela »*. Si l'originalité de son concept a été appréciée, le fait qu'il ne soit pas rattachable à un genre vidéoludique connu et que ses mécanismes présentent de légers défauts d'équilibrage a clairement joué en sa défaveur. Concrètement, le seul membre du jury du concours à avoir apprécié ce jeu au point de vouloir lui décerner un prix était un Game Designer professionnel. De par son métier et sa grande culture vidéoludique, il a apprécié l'aspect novateur de l'approche « gameplay expérimental » de ce Serious Game, qui permet de proposer un principe de jeu innovant bien qu'imparfait. Les autres membres du jury ont plutôt été perturbés par l'originalité du concept. Ils n'ont donc pas cherché outre mesure à analyser le fonctionnement du jeu pour gagner, passant ainsi à côté de son message. Ils lui ont préféré des jeux au gameplay plus « classique » car se rattachant à un genre existant *(en l'occurrence les jeux d'arcade de type « course d'obstacle »)*, mais dont les qualités ludiques du principe de jeu sont éprouvées par les nombreux autres jeux du genre. Précisons tout de même que, par manque de temps, nous n'avons pas été en mesure d'implémenter un véritable « tutorial » autant adapté aux novices que ceux d'*Eco-Reporter (p.122)* ou du *Jardinier Ecolo (p.126)*. Le fait que les mécanismes de jeux soient exposés de manière relativement sommaire contribue sûrement à expliquer pourquoi le jeu n'a touché que le « joueur expert » qu'est un Game Designer et non les autres membres du jury qui étaient moins familiers avec la culture vidéoludique.

Quoi qu'il en soit, le fait de reprendre ou de s'inspirer de mécanismes de jeux connus et éprouvés facilite clairement le processus de conception d'un Serious Game par rapport à l'invention de nouveaux principes de jeu. Les phases de tests et de corrections peuvent alors se focaliser sur la cohérence entre les dimensions « sérieuse » et « ludique ». De son côté, l'approche « gameplay expérimental » implique en outre de passer par une phase préalable visant à éprouver longuement la qualité purement ludique du jeu. Si cela est loin d'être impossible, cette approche était visiblement trop ambitieuse pour les quatre journées qui ont été consacrés au projet *Blabbing Blobs*. Nous retiendrons quand même de cette expérience qu'il existe d'autres voies que le recours à des genres vidéoludique connus pour créer des Serious Games, mais que l'approche « gameplay expérimental » présente des défis plus importants pour le concepteur.

2.9 SYNTHESE : DIFFUSER UN CONTENU SERIEUX A TRAVERS LE JEU VIDEO

Ces trois expériences concrètes nous amènent de nombreux enseignements relatifs aux spécificités de la conception de Serious Games. Tout d'abord, elles permettent d'appliquer le *modèle générique DICE (p.103)* du processus de conception d'un Serious Game. Si les étapes suivies pour la création d'*Eco-Reporter (p.119)*, du *Jardinier Ecolo (p.123)* et de *Blabbing Blobs (p.140)* sont loin d'être absolument identiques, toutes s'inscrivent dans les quatre étapes génériques : *définir, inventer, créer* et *évaluer*. Le processus itératif de cette méthodologie de conception est le principal moteur de la création de ces Serious Games, qui s'est à chaque fois appuyée sur la réalisation de nombreux prototypes pour autant de tests utilisateurs.

Ce retour d'expérience montre également qu'il existe plusieurs approches de la conception de Serious Games « intrinsèques » *(p.106)*. Nous pouvons d'ailleurs ajouter à ce retour d'expérience le projet *geDriver (p.86)*, lui aussi basé sur une approche « intrinsèque » mais destiné à prodiguer un entraînement au lieu de diffuser un message. Nous observons alors

deux types d'approches pour créer un Serious Game mélangeant les dimensions « sérieuse » et « ludique ». La première consiste à s'inspirer d'un genre vidéoludique existant pour concevoir un principe de jeu intégrant le contenu sérieux, comme illustré par *Eco-Reporter*, *Le Jardinier Ecolo* et *geDriver*. Dans cette approche, le concepteur part de la dimension « ludique » pour aller vers le contenu « sérieux ». En effet, le contenu sérieux est analysé et retravaillé de manière à être adapté à des principes de jeux éprouvés, comme nous l'avons par exemple fait en nous limitant à la diffusion d'un seul « message » pour respecter le côté « action » du *Jardinier Ecolo* *(p.118)*. Bien entendu, il reste possible d'adapter les codes d'un genre vidéoludique pour un contenu sérieux donné, comme l'illustre la manière dont avons ajouté la rédaction d'un article au genre du « jeu d'aventure » avec *Eco-Reporter* afin de pousser le joueur à s'intéresser aux messages du jeu *(p.119)*. La seconde approche vise à inventer un nouveau principe de jeu en partant du contenu sérieux. Dans cette approche de « gameplay expérimental », le concepteur de jeu aura donc plutôt tendance à ajuster ses mécanismes de jeu par rapport au contenu sérieux et non l'inverse. Mais l'approche visant à inventer des concepts de jeux innovants n'est pas neutre quant au public susceptible d'être intéressé par le jeu résultant, comme l'illustre *Blabbing Blobs* *(p.134)*. Ainsi la création d'un « gameplay expérimental » suscitera plus facilement l'intérêt d'un public possédant une culture vidéoludique solide, en lui proposant un jeu sortant des canons habituels. Si elle est mal maîtrisée, cette approche pourra potentiellement dérouter des novices en la matière. Si ces derniers ne sont pas forcément à la recherche de jeux « classiques » et peuvent tout à fait être intéressés par un jeu expérimental, ce jeu devra néanmoins redoubler d'efforts pour expliquer son fonctionnement à un public peu habitué au jeu vidéo. C'est d'ailleurs là que nous voyons tout l'intérêt des approches de type « tutorial » pour le Serious Game, car elles semblent permettre d'expliquer des jeux relativement complexes à un public n'ayant jamais touché à un jeu vidéo *(p.126)*.

Au final, que ce soit pour la diffusion de messages ou pour prodiguer un entraînement, nous retenons de l'expérience de conception de ces quatre projets qu'il existe deux principaux vecteurs de diffusion d'un contenu sérieux par un Serious Game « intrinsèque » :

- **Par le « contenu du jeu »**. Par exemple, *Eco-Reporter* met le joueur dans une situation qui le pousse à s'intéresser à un contenu sérieux. Pour gagner, il faut écrire un article. Pour écrire un article, il faut organiser de nombreuses informations, et donc passer du temps à les lire. De plus, pour obtenir ces informations, il faut auparavant explorer longuement le quartier, et donc se confronter à sa géographie. Logiquement, après une partie d'*Eco-Reporter* le joueur est sensé avoir identifié les messages délivrés, car ces derniers sont explicitement matérialisés dans le jeu qui propose une sorte de « quête » pour les acquérir. Le fait que le contenu sérieux soit présenté de manière directe facilite a priori sa diffusion, et permet même d'envisager la diffusion de nombreuses informations au sein d'un seul jeu. C'est sans doute pour cette raison que le genre du « jeu d'aventure » a été jugé comme *« particulièrement pertinent pour la diffusion de contenu pédagogique »* par une équipe de chercheurs *(p.119)*.

- **Par les « règles du jeu »**. *Le Jardinier Ecolo*, *geDriver* et *Blabbing Blobs* sont tous les trois basés sur une logique où le joueur doit découvrir, par essais et erreurs, une « stratégie de jeu » lui permettant de gagner. Le contenu sérieux est cette fois-ci intégré directement au sein de cette stratégie gagnante, qu'il s'agisse des messages sur le type de plantes les plus « écocitoyennes » ou sur les gestes de conduite permettant de limiter le rejet de *CO2*. Le contenu sérieux n'est donc plus présenté de manière « explicite » au joueur, qui doit faire un effort cognitif conséquent d'analyse des règles du jeu pour arriver à identifier la marche à suivre pour gagner. Ce faisant, il décryptera le contenu sérieux délivré par le jeu. Le concepteur du jeu est donc parfois obligé de recourir à des méthodes telles que la *« rhétorique de l'échec »* *(p.132)*, dans laquelle le

jeu fait sciemment perdre le joueur pour le pousser à réfléchir sur les raisons de son échec. Il l'incite ainsi à analyser les mécanismes du jeu pour prendre conscience du contenu sérieux qu'il véhicule.

Si ce dernier vecteur semble a priori plus complexe à mettre en œuvre que le premier, il faut bien comprendre que ces deux vecteurs ne se retrouvent pas dans les mêmes genres de jeux. Ainsi, la diffusion d'un contenu sérieux par le « contenu du jeu » va plutôt se limiter aux jeux de « réflexion », et plus particulièrement aux jeux d'aventure qui reposent sur des scénarios très riches. A contrario, la diffusion d'un contenu sérieux par les « règles du jeu » est plus adaptée aux titres dans lesquels l'action prime sur la réflexion, ou en tout cas aux jeux dont le système de règles est plus riche que le scénario. Cela nous renvoie à la différence entre *« jeux de progression »* et *« jeux d'émergence »* introduite par ***Juul*** (2005). D'après ce chercheur, deux catégories de jeux vidéo s'opposent. D'un côté, les *« jeux de progression »* reposent sur l'ajout de nouveaux éléments pour maintenir l'intérêt du joueur tout au long de la partie. Ces ajouts ont lieu selon un scénario défini par le concepteur, ce qui signifie que ces jeux peuvent être « terminés » une fois que tous les éléments ont étés découverts. Le chercheur cite les jeux d'aventure comme exemple de *« jeux de progression »*. A l'inverse, les *« jeux d'émergence »* sont uniquement basés sur des règles qui interagissent pour créer sans cesse de nouvelles situations de jeux intéressantes. Un *« jeu d'émergence »* comme *Tetris* ***(Alexey Pajitnov & Vadim Gerasimov, 1985)*** peut donc potentiellement générer à l'infini des nouvelles situations de jeu, voire même des situations qui n'ont pas forcément été explicitement imaginées par le concepteur du jeu. Dans la pratique ces deux formes vidéoludique se trouvent rarement à « l'état pur », car de nombreux titres mélangent les concepts *« d'émergence »* et de *« progression »*. Mais le fait de tendre plutôt vers l'un ou l'autre de ces deux types de jeux vidéo permet d'attirer un public de joueurs différents. En effet, nous avons vu qu'il existe différents profils de joueurs *(p.71)*, et que certains joueurs peuvent n'éprouver aucun intérêt particulier pour un type de jeu donné *(p.130)*. Afin de pouvoir créer des Serious Games « intrinsèques » s'adressant à un large public, il semble donc important pour un concepteur de maîtriser les deux vecteurs de diffusion de contenu que nous venons d'identifier. En effet, le vecteur de diffusion par le « contenu du jeu » permet de créer des Serious Games reposant sur le type *« jeu de progression »* alors que le vecteur de diffusion par les « règles du jeu » sera quant à lui la base d'un Serious Game relevant plutôt du type *« jeu d'émergence »*. Et dans le cas d'un Serious Games reposant à la fois sur *« l'émergence »* et *« la progression »*, l'utilisation conjointe des deux vecteurs de diffusion d'un contenu sérieux semble tout à fait indiquée.

Comme nous venons de le voir, les divers enseignements que nous tirons de ce retour d'expérience ne sont pas voués à rester au stade empirique. Il semble possible de les mettre en perspective au travers de travaux théoriques issus de la recherche scientifique sur les jeux vidéo, comme nous proposons de continuer à le faire dans la section suivante.

3 APPROFONDISSEMENT DE CE RETOUR D'EXPERIENCE

Les enseignements empiriques que nous tirons du retour d'expérience de la réalisation de quatre projets de Serious Games sont loin d'être des constatations isolées. A travers la littérature scientifique dédiée au jeu vidéo, nous pouvons trouver d'autres travaux portant sur les thématiques que nous venons d'aborder, qui permettent donc d'approfondir certaines de nos réflexions.

Comme nous l'avons constaté à travers la création des Serious Games pour *Andromède* *(p.120 et p.126)*, l'approche « intrinsèque » de conception d'un Serious Game simplifie grandement l'évaluation de la transmission du contenu sérieux. En effet, puisque « gagner le jeu » implique d'avoir assimilé le contenu sérieux diffusé par le Serious Game, il suffit généralement de suivre l'évolution du joueur dans le jeu pour évaluer sa réception du contenu sérieux. Mais le suivi des joueurs lors d'une partie de jeu vidéo est loin d'être une question triviale, et se trouve même être un thème de recherche actuel. Les travaux proposés sont applicables à différents types de contexte d'utilisation, c'est-à-dire à différents environnements logiciels, tel que détaillé ci-dessous.

3.1.1 LE SUIVI DU JOUEUR PAR L'INTERMEDIAIRE D'UN LMS

Les environnements de type *LMS*[38] sont généralement utilisés dans un contexte pédagogique, que ce soit dans un cadre scolaire ou dans le domaine de la formation professionnelle. Dans un *LMS*, le Serious Game est considéré comme une « activité pédagogique ». Tout *LMS* qui se respecte est capable d'enregistrer les données renvoyées par les différentes « activités pédagogiques » qu'il propose, pour peu que ces données respectent une norme précise, en général le standard *SCORM*. L'intégration de Serious Games à un *LMS* par le respect de ce standard a déjà été étudiée, et se trouve être une fonctionnalité présente dans plusieurs outils techniques dédiés au Serious Game, tels que *Thinking Worlds* *(p.214)*, *SGTools* *(p.214)* ou encore *<e-Adventure>* *(p.222)*.

Cependant, des travaux récents montrent les limites de la norme *SCORM*, au départ établie pour suivre les résultats de « quiz » pédagogiques. Cette norme n'est pas adaptée à l'enregistrement des nombreuses données qui peuvent être renvoyées par une application plus complexe telle qu'un Serious Game. Par exemple, ***Thomas*** (2010) met en évidence trois limitations : l'impossibilité d'enregistrer des indicateurs sur les types d'erreurs commises par l'apprenant, l'enregistrement limité à la dernière (ou meilleure) tentative de l'apprenant interdisant les analyses différentielles entre plusieurs « stratégies de jeu » dans un Serious Game et l'impossibilité de partager les résultats entre différents apprenants qui empêche la création de « tableaux de meilleurs score » pouvant être un élément de motivation fort pour un Serious Game *(p.132)*. Deux types de solutions sont proposés pour palier à ces limites. La première consiste à s'appuyer des *LMS* reposant sur des normes plus ouvertes ou plus adaptées que *SCORM*, à l'image d'une expérimentation faite sur *IMS-LD* (Burgos et al., 2008). L'autre solution, proposée par ***Thomas***, est de recourir à une base de donnée externe au *LMS* pour enregistrer toute donnée renvoyée par le Serious Game sans contraintes particulières. Cette dernière solution relève donc plutôt de la catégorie *« suivi du joueur par l'intermédiaire d'un outil personnalisé »* que de l'utilisation d'un *LMS* pour ce même suivi.

3.1.2 LE SUIVI DU JOUEUR PAR L'INTERMEDIAIRE D'UN OUTIL PERSONNALISE

Le suivi du joueur a ici lieu dans un environnement logiciel directement contrôlé par le créateur du Serious Game. Il est donc libre d'utiliser la méthode qu'il souhaite pour suivre le parcours du joueur dans son jeu. Plusieurs approches très différentes ont été proposées.

La première d'entre elles est très simple : elle consiste à filmer un jeu en train d'être joué par un joueur, puis d'analyser l'enregistrement vidéo. On ne filme pas le joueur, mais on enregistre tout simplement ce qui est affiché sur son écran. Cette méthode d'analyse n'est donc applicable qu'à un contexte qualitatif, mais elle est intéressante car elle introduit un

[38] Acronyme de **L**earning **M**anagement **S**ystem. Désigne un système logiciel destiné à accompagner et suivre des apprenants dans un cursus de formation. Un LMS rassemble plusieurs outils (cours, exercices, chat, forum…) et enregistre leur utilisation (notes obtenues aux exercices…) pour chaque apprenant, facilitant ainsi leur suivi dans le cadre de formations à distance.

modèle pour l'analyse d'une partie de jeu vidéo : la *Gameplay Video Segmentation Method* (Neubauer, 2006). Concrètement, elle permet de diviser l'enregistrement d'une partie de jeu vidéo selon cinq niveaux de lecture : les *« éléments primaires »*, les *« actions du joueur »*, la *« partie de mission »*, la *« mission »* et le *« jeu complet »*. Les *« éléments primaires »* sont les plus petites interactions observables lors d'une partie, et correspondent globalement à chacune des actions du joueur. Plusieurs de ces *« éléments primaires »* permettent de composer une *« action du joueur »*, qui correspond à quelque chose de plus large, par exemple tous les mouvements qu'un joueur fera pour aller chercher un bonus ou pour tuer un ennemi. Tous les éléments d'un niveau de lecture donné peuvent être regroupés pour créer un niveau de lecture supérieur. Ce modèle défini ainsi une hiérarchie de cinq niveaux de lecture permettant d'analyser plus ou moins finement comment un joueur utilise un jeu. Une approche similaire a été utilisée pour suivre le comportement du joueur en temps réel durant le jeu, et non pas à posteriori à partir d'enregistrements vidéo. Dans sa thèse, ***Levieux*** (2011) propose une méthode permettant de mesurer la difficulté des jeux vidéo. Pour cela, il a réalisé un logiciel suivant le joueur en temps réel. Il utilise alors les données enregistrées pour évaluer la difficulté du jeu, dans l'optique de pouvoir l'adapter automatiquement par la suite. Ce système d'analyse s'appuie sur un modèle théorique de segmentation d'une partie de jeu vidéo. Ce modèle découpe un jeu en *« challenges »* qui correspondent à un objectif simple, comme *« tuer un ennemi avec un couteau »*. Le joueur peut soit réussir, soit échouer ces *« challenges »*. Ils proposent donc un résultat binaire facile à enregistrer. Les *« challenges »* de base sont dénommés *« challenges atomiques »*, car plusieurs d'entre eux peuvent être associés pour former des *« challenges composites »*. Il est ainsi possible de découper une session de jeu en une hiérarchie de *« challenges »*. Pour réussir un *« challenge »*, un joueur doit faire preuve d'une certaine *« capacité »*. Cette notion de *« capacité »* va alors être définie au cas par cas selon les jeux, afin de permettre le suivi du joueur. Par exemple, dans un jeu de tir en vue subjective, une des *« capacités »* mesurée sera *« tirer »*. Pour analyser si le joueur possède la capacité de *« tirer »*, il suffit tout simplement d'étudier s'il réussi le challenge *« toucher un ennemi »* à chaque fois qu'il appuie sur le bouton de tir. Le suivi du joueur est donc ici fait en enregistrant continuellement sa réussite à des *« challenges »* qui auront été défini explicitement par le concepteur. Contrairement à la méthode de ***Neubauer***, le système de ***Levieux*** est purement logiciel. Il pourrait donc tout à fait être utilisé pour évaluer l'impact d'un Serious Game, pour peu que le concepteur du jeu configure ce système pour mesurer des *« challenges »* pertinents.

Mais la méthode la plus courante consiste tout simplement à programmer directement à l'intérieur d'un jeu des méthodes d'enregistrement de données qui sont ensuite stockées dans une base de donnée externe pour être analysée. ***Thomas*** (2010) a utilisé cette méthode pour suivre les joueurs du Serious Game *Starbank* ***(KTM Advance, 2009)***. Dans un registre similaire, *Refraction* ***(Erik Andersen & al., 2010)*** est associé à un outil de suivi et d'analyse des joueurs appelé *Playtracer* (Andersen, Liu, Apter, Boucher-Genesse, & Popović, 2010). Cet outil s'appuie sur la définition du jeu, en tant qu'objet, comme un *« système à état variable »* *(p.156)*. Chaque niveau du jeu est donc défini par le système d'analyse comme une collection « d'états », tels que *« début du niveau »*, *« pièce X mal placée »*, *« laser Y entrant du bon côté »* ou encore *« niveau terminé »*. Le système enregistre ensuite, pour chaque joueur, les différents « états » du jeu qu'il va atteindre, et le cheminement qu'il a parcouru pour les atteindre. Pour chacun des niveaux de jeu, *Playtracer* est capable de produire un graphique avec des cercles représentant les divers « états » du jeu. Plus un « état » a été « visité » par les joueurs, plus le cercle sera gros. Mais ce graphique représente également les transitions entre les états : plus des cercles sont proches, plus le nombre de joueurs passant directement de l'un à l'autre des « états » qu'ils représentent est élevé. Cette méthode permet une analyse graphique du parcours des joueurs, facilitant l'interprétation des données collectées lors du suivi. Si ces deux exemples sont liés à des Serious Games précis, il est également possible d'intégrer de telles fonctionnalités à un outil technique de création de

Serious Games, permettant ainsi à tous les jeux créés avec cet outil d'en bénéficier. Par exemple, *Virtuoso* permet d'enregistrer des données définies par le concepteur du jeu dans une base de donnée de type *MySQL* *(p.220)*. De même, tous les jeux créés à partir d'une *Coquille Générique de Jeu Educatif* enregistrent automatiquement plusieurs variables de suivi des joueurs, sans même que le concepteur du jeu ait à s'en préoccuper *(p.227)*.

3.1.3 LE SUIVI DU JOUEUR PAR L'INTERMEDIAIRE D'UN OUTIL GENERIQUE

Un troisième cas de figure peut se présenter lors de la distribution d'un Serious Game au grand public par le biais d'Internet. Au lieu d'être centralisé sur un site bien déterminé *(en général celui du commanditaire)*, il peut parfois être pertinent de le diffuser sur les nombreux sites Internet de type « portails de jeux » tels que *Kongregate* ou *Newgrounds* afin de toucher un maximum de joueurs. Dans ce cas, la diffusion du jeu a lieu dans un environnement qui n'est pas directement contrôlé par le concepteur du jeu, ce qui peut avoir des conséquences sur les techniques utilisées pour suivre les joueurs. Certes, dans la plupart des cas il restera toujours possible d'enregistrer les données par Internet dans une base de données, au prix des quelques ajustements techniques qui s'imposent *(ouverture des connexions extérieures à la base de donnée donc renforcement de la sécurité d'accès, dimensionnement du serveur pour tenir une charge de connexions simultanées extrêmement élevée…)*. Mais il existe une autre solution adaptée à ce type de situation : l'utilisation des systèmes de mesure d'audience utilisés par les concepteurs de jeux vidéo de divertissement. En effet, ces derniers ont également besoin de mesurer l'intérêt des joueurs pour leurs créations, afin de s'assurer que le jeu est « fun ». Le fait d'essayer ainsi de mesurer le plaisir de jeu éprouvé par les joueurs s'appelle avoir recours à des *« metrics »* pour la conception (Cook, 2009). Il semble donc tout à fait possible de « détourner » ces systèmes de mesure pour essayer d'évaluer l'impact de Serious Games, en tout cas pour les jeux conçus selon une approche « intrinsèque ». Par exemple, face à l'éventualité d'une diffusion large de *Eco-Reporter* et *Le Jardinier Ecolo* sur plusieurs sites Internet, nous avons expérimenté deux outils de *« metrics »* pour récolter des données d'évaluation quantitatives, comme détaillé précédemment *(p.134)*. Ces outils nous ont permis d'enregistrer tout type de données, à l'image des outils personnalisés. Ils proposent même des modes de visualisation suffisamment détaillés pour permettre des analyses aussi poussées qu'avec *Playtracer* *(p.135)*. Et pourtant, leur mise en place fut aussi aisée que la publication d'un jeu sur *LMS* : une simple *API* à intégrer au jeu.

Bien qu'elle semble relativement peu utilisée dans le champ du Serious Game, l'approche qui vise à suivre le joueur par le biais d'un outil générique nous semble donc être très prometteuse. A partir de notre retour d'expérience avec ce type d'outil, nous observons qu'il allie la facilité de mise en place d'un *LMS* avec la liberté de définition des critères de suivis d'un outil personnalisé. Si les outils génériques de suivi présentent certes des limites et contraintes propres, ils semblent donc constituer un excellent compromis.

3.2 EXPLIQUER UN JEU N'EST-IL PAS AUSSI IMPORTANT QUE LE JEU LUI-MEME ?

Un autre enseignement primordial que nous retirons du retour d'expérience sur la conception des Serious Games sur *Andromède* est l'importance d'expliquer le fonctionnement du jeu aux utilisateurs. En effet, de nombreux projets de Serious Games s'adressent à un public qui n'est pas forcément familier avec le jeu vidéo *(p.18)*, quand ils ne se destinent pas à des personnes n'ayant jamais joué à un jeu vidéo de leur vie. Il semble donc primordial de trouver des moyens permettant d'expliquer à ces joueurs novices comment utiliser le jeu si l'on souhaite leur transmettre un contenu sérieux à travers ce dernier.

Contrairement aux apparences, le secteur du jeu vidéo de divertissement n'est pas forcément le plus pertinent pour tenter de répondre à cette problématique. En effet, l'industrie du jeu vidéo de divertissement s'adresse à un public de « spécialistes », comme nous pouvons le

constater à travers la catégorisation de jeux en « genres » qui renvoient à des « codes » liés au principe du jeu. Les concepteurs de jeux vidéo partent du principe qu'au fur et à mesure que les joueurs s'adonnent à un « genre » de jeu donné, ils apprennent à y jouer *(p.111)* et deviennent de plus en plus performants. Les concepteurs ont donc naturellement tendance à complexifier leurs titres au fur et à mesure de l'évolution de l'histoire du jeu vidéo. Les tutoriaux de ces jeux auront donc tendance à prendre pour acquises les « bases » du genre, et à n'expliquer que les « nouveautés » introduites par un titre donné. Mais comme nous l'avons constaté *(p.122)*, ces fameuses « bases » des genres de jeu vidéo ne sont absolument pas maîtrisées par des joueurs novices, à qui il convient donc d'expliquer intégralement comment manipuler le jeu.

Pour autant, ce type de questionnement n'est pas cantonné au domaine du Serious Game, et se voit incarné par un « courant » du jeu vidéo de divertissement : le *« Casual Game »* (Juul, 2009). Ce terme désigne une vague de jeux vidéo destinés à un public qui n'est pas forcément passionné par le jeu. Il se caractérise donc par des principes de jeux *« simples à comprendre mais difficiles à maîtriser »* (Kremer, 2009). De plus, ces jeux sont pensés pour être utilisés lors de sessions de jeu « courtes », et doivent potentiellement s'adresser à un public plus varié en terme d'âge et de sexe que la tranche des *« joueurs masculins de 12 à 25 ans »* traditionnellement au cœur du marché de l'industrie du jeu vidéo. Incarnée par des titres comme *Bejeweled* ***(Popcap Games, 2001)***, *Diner Dash* ***(GameLab, 2003)***, *Wii Sports* ***(Nintendo, 2006)*** ou encore *Angry Birds* ***(Rovio Mobile, 2009)***, la vague du *« Casual Game »* a permis au jeu vidéo de divertissement de s'adresser à un public plus large (Rohrl, 2008). Le fait de viser un public plus varié et moins spécialiste en jeu vidéo implique que les Game Designers doivent imaginer des concepts de jeux différents pour plaire à cette nouvelle audience (Trefry, 2010). Par exemple, ***Jason Kapalka*** (2006), directeur créatif et co-fondateur du studio ***PopCap Games*** livre ses *« 10 façons de créer un mauvais Casual Game »*. De manière ironique, cet article livre dix « conseils de conception » *(p.80)* mettant en lumière les spécificités du *« Casual Game »* par rapport aux autres jeux vidéo de divertissement. Le conseil numéro huit, *« Attendre de vos utilisateurs qu'ils lisent »*[39] renvoie directement aux problèmes que nous avons rencontrés lors de la recherche d'un moyen permettant d'expliquer *Le Jardinier Ecolo* *(p.126)*. ***Kapalka*** expose ici l'importance d'un « tutorial interactif » intégré au jeu, et affirme qu'il est illusoire d'imaginer que des joueurs novices vont lire de nombreuses pages de textes, aussi bien écrites soit-elles, pour découvrir le fonctionnement du jeu.

Les problématiques du *« Casual Game »* sur l'explication du jeu renvoient donc de manière globale à des problématiques liées au Serious Game. Nous pouvons par exemple penser au secteur du « ludo-éducatif », dont la notion de « consigne » pour expliquer les différentes activités aux apprenant est primordiale, et détermine en grande partie le potentiel pédagogique réel d'un tel titre (Kellner, 2007). Mais, étant donné que *« tous les jeux impliquent un apprentissage »* *(p.111)*, la question de l'explication du jeu est donc finalement transversale à tout jeu vidéo existant. Comme pour l'éducation, les approches déployés pour expliquer « comment jouer » différent en fonction du niveau de connaissance préalable des joueurs visés. Etant donné qu'il s'adresse à un public qui n'est pas forcément composé de joueurs experts, le champ du « Serious Game » semble donc pouvoir grandement bénéficier des techniques développées par le courant du *« Casual Game »* en matière d'explication du fonctionnement du jeu. Cette remarque sur l'importance d'un « tutorial » permettant d'expliquer de manière progressive et interactive « comment jouer » à des novices ne se limite d'ailleurs pas aux jeux. En effet, comme nous avons pu le constater lors du retour d'expérience de ***Julien Llanas*** sur l'utilisation de *LittleBigPlanet* en contexte scolaire

[39] *"Expect users to read"*

(p.206), le « tutorial » semble également être un excellent moyen pour apprendre à des utilisateurs à manipuler un outil technique de création vidéoludique.

3.3 « GAME DESIGN » ET « PLAY DESIGN »

Enfin, l'expérience de conception de ces projets de Serious Game semble faire ressortir deux « niveaux » de travail. Tout d'abord, le concepteur doit travailler le jeu « en tant qu'objet », en créant des règles de jeu et autres composantes d'un jeu vidéo *(p.156)*. Une fois ce travail effectué et validé par des premiers prototypes, une phase relativement longue « d'affinage » démarre *(p.63)*. Le concepteur doit alors s'assurer que l'expérience de jeu globale est satisfaisante pour l'utilisateur, et que le contenu sérieux est bien délivré, toujours à partir de cycles itératifs de prototypage et d'évaluation. Ces deux « niveaux » nous renvoient en fait aux angles définitoires proposés par ***Juul*** *(p.43)*, pour qui la définition du jeu peut être entendue en tant qu'objet *(« The Game »)* ou comme la relation entre un jeu et un joueur *(« The Player »)*. Cela fait également écho à certains modèles formels de la relation joueur-jeu tel que le *modèle MDA (p.75)*, qui distingue le niveau des *« Mechanics »* dédié à l'objet-jeu et le niveau des *« Aesthetics »* pour les émotions ressenties par le jeu. Nous retrouvons également cette distinction dans certains ouvrages de référence sur le Game Design (Fullerton, 2008; Schell, 2008b). Si ces auteurs rappellent qu'au final un concepteur se doit d'œuvrer pour proposer la meilleure expérience ludique possible au joueur, ils précisent que ce travail se fait de manière itérative en plusieurs étapes. Il faut donc commencer par créer une base de jeu fonctionnelle avant de pouvoir l'affiner en vue de délivrer une expérience particulière au joueur. Loin d'être anodine, cette distinction semble fondamentale pour les concepteurs de jeu vidéo car elle détermine leurs critères d'évaluation. En effet, lorsqu'il se focalise sur la construction du jeu en tant qu'objet, le concepteur cherchera principalement à valider le bon fonctionnement des divers mécanismes de jeu. Une fois cette base fonctionnelle, il prendra alors en compte de manière plus précise l'expérience que retirera le joueur de l'utilisation de son jeu. En référence à un concept esquissé par ***Genvo*** (2008), nous proposons d'appeler la phase dédiée à la création de « l'objet-jeu » le *« Game Design »* et la phase dédiée à l'expérience de jeu globale le *« Play Design »*. Nous introduisons donc ici une quatrième définition possible pour le terme *« Game Design » (p.54)*. Au-delà de l'aspect créatif, ces deux phases se distinguent également par des besoins différents en terme d'outils théoriques. En effet, lors de la création de « l'objet-jeu », le travail d'évaluation et de prototypage peut s'appuyer sur des modèles théoriques formels qui permettent de modéliser la structure du jeu *(p.68)*. Par contre, lors du *« Play Design »*, le concepteur va chercher à susciter différentes émotions au joueur interagissant avec son jeu. Son modèle de référence pour les évaluations ne sera donc plus une modélisation formelle, mais le ressenti de joueurs humains qui aura été « évalué » selon différentes méthodes *(p.129 et p.134)*.

Nous pouvons illustrer ces deux « phases » de la conception grâce au Serious Game *Eco-Reporter (p.119)*. Un des premiers prototypes destiné à éprouver le système de dialogues était dépourvu de toute forme de « transition » entre les différents écrans du jeu. Ainsi, lorsque le joueur cliquait sur une question pour interroger un personnage, le texte de réponse du personnage apparaissait immédiatement et dans son intégralité à l'écran. De plus, si cette réponse était associée à une « note » que le joueur peut enregistrer, une petite fenêtre proposant de noter l'information apparaissait simultanément. Lors des tests menés sur ce prototype, nous avons constaté que les gens avaient tendance à ne pas lire le texte affiché à l'écran, particulièrement pour les dialogues qui donnait une information à noter *(donc ceux qui diffusent le contenu sérieux)*. Pourtant, les « règles du jeu » étaient conformes à ce que nous avions imaginé et formalisé lors de l'étape *« Imaginer »*. Si le principe interactif des dialogues était parfaitement fonctionnel, en revanche l'expérience de jeu qu'il délivrait en l'état n'était pas satisfaisante. En nous appuyant sur les retours des utilisateurs nous avons alors rajouté des « transitions ». Tout d'abord, une fois que le joueur clique sur une question,

le texte de la réponse du personnage est affiché de manière progressive, ligne par ligne. Une fois cette affichage terminé, un temps d'attente de quelques secondes s'écoule avant que la fenêtre prévenant le joueur qu'il peut noter cette information apparaisse. Le simple ajout de ces deux transitions a permis de corriger le problème rencontré. Dorénavant, les joueurs prennent le temps de lire les réponses qui apparaissent avant de les « noter ». De plus, nous constatons que ce changement ralenti le rythme général du jeu, et aide ainsi les joueurs à se mettre dans une posture « calme » plus favorable à lecture des dialogues du jeu. Cet exemple illustre comment nous avons tout d'abord créé des prototypes destinés à éprouver le *« Game Design »* avant de nous concentrer sur l'expérience globale délivrée par le jeu en nous focalisant sur le *« Play Design »*.

4 SYNTHESE : UNE APPROCHE PERSONNELLE DE LA CONCEPTION DE SERIOUS GAME

Fort de ce retour d'expérience lié au *CIFRE* et de l'étude des outils théoriques de conception de jeux vidéo et de Serious Games, nous souhaitons formaliser la méthodologie de conception que nous avons utilisée pour créer les quatre projets de Serious Games présentés précédemment. Cette méthodologie a également été influencée par tous les textes traitant de Game Design que nous avons analysés *(p.59)*, en particulier l'ouvrage de ***Fullerton*** (2008). Cette méthodologie s'articule autour d'une « série d'étapes » du processus de conception, étayée par les divers outils présentés dans cette thèse. Précisons que, comme le rappelait ***Crawford*** *(p.84)*, cette méthodologie n'a bien évidement pas vocation à faire consensus. Partant du constat qu'une « méthodologie universelle » de conception de Serious Game n'existe pas *(p.103)*, nous souhaitons modestement formaliser celle que nous avons élaborée suite au travail de recherche exposé dans cette thèse. Chaque concepteur reste alors libre de ne pas la trouver à son goût, ou de s'en inspirer pour créer sa propre méthodologie de « Serious Game Design ». Nous proposons la série d'étapes suivante, basée sur le *modèle générique DICE (p.103)* :

1) Définir le contenu sérieux

- Formalisation du « contenu sérieux » à transmettre à travers le jeu avec les « experts » éventuellement associés au projet *(commanditaires, enseignants…)* ou suite à des recherches documentaires sur le thème à traiter.
- Définition du public visé par le Serious Game : âge, sexe, niveau de connaissance préalable du contenu sérieux et niveau d'expérience préalable avec le jeu vidéo.
- Formalisation de critères d'évaluation pour la pertinence du Serious Game. En accord avec les éventuels « experts » associés au projet, il s'agit de définir la mission exacte du Serious Game *(délivrer un message précis, prodiguer un entraînement sur telle procédure…)* qui sera ensuite utilisée pour évaluer sa pertinence.
- Analyse de l'existant : déconstruction analytique de Serious Games éventuellement existants sur le sujet *(p.112)*.

2) Imaginer un principe de jeu

- Choix d'une approche d'association des dimensions « sérieuse » et « ludique » *(p.106)* :
    - Approche *« extrinsèque »* : le contenu sérieux et le jeu ne sont pas forcément mélangés, la partie « jeu » étant utilisée comme une « récompense » suite à la réussite des phases « sérieuses » de l'application.
    - Approche *« intrinsèque »* : le contenu sérieux est intégré au cœur même du principe de jeu. Les deux dimensions ne sont donc plus dissociables et s'enrichissent mutuellement.

- Invention d'un principe de jeu. Plusieurs approches créatives sont possibles :
    - Approche *« jeu de genre »* : la création d'un principe de jeu s'inspire d'un genre vidéoludique existant. Le concepteur sélectionne le genre le plus adapté au contenu sérieux qu'il souhaite transmettre, et adapte éventuellement les codes du genre pour servir la vocation utilitaire du jeu vidéo. Selon le genre de jeu choisi, différents « vecteurs » de diffusion d'un contenu sérieux sont disponibles pour un Serious Game basé sur une approche *« intrinsèque »* *(p.142)*.
    - Approche *« gameplay expérimental »* : la création d'un principe de jeu est complètement originale. Cette approche permet de partir du contenu et de construire librement un principe de jeu qui lui soit adapté. Cependant, elle nécessitera de plus nombreux cycles de prototypages et d'évaluations afin de s'assurer de l'intérêt ludique du principe de jeu ainsi inventé, aucun jeu du même genre n'étant là pour servir de référent. Les différents « vecteurs » de diffusion d'un contenu sérieux restent généralement applicables *(p.142)*.
- Formalisation du principe de jeu ainsi inventé. Cette formalisation doit détailler aussi bien les mécanismes et règles du jeu *(« Game Design »)*, que la teneur de l'expérience ludique souhaitée *(« Play Design »)*. Divers outils théoriques peuvent être utilisés à ces fins.

3) Créer un jeu vidéo fonctionnel

- Réalisation de prototypes de plus en plus complets. Si les premiers prototypes sont destinés à tester uniquement certains aspects du jeu, ils finiront par permettre d'évaluer le principe de jeu dans son intégralité, avant de permettre d'ajuster de nombreux petits détails pour délivrer la meilleure expérience possible au joueur.
- La réalisation de ces prototypes peut éventuellement s'appuyer sur un outil technique adapté. Le choix d'un tel outil dépendra du principe de jeu formalisé à l'étape précédente, des compétences techniques du concepteur, ainsi que du temps et du budget dont il dispose.

4) Evaluer la pertinence du jeu

- Evaluation des prototypes pour s'assurer qu'ils correspondent aux attentes du concepteur. Les premières évaluations visent à vérifier l'adéquation entre les attentes formalisées lors de l'étape *« Imaginer »* et les jeux réalisés dans l'étape *« Créer »*. Une fois ces phases d'évaluation validées, il faut vérifier que le Serious Game ainsi obtenu remplit bien sa mission en l'évaluant grâce aux critères formalisés lors de l'étape *« Définir »*.
- Tant que l'évaluation ne produit pas des résultats satisfaisants, le concepteur revient à l'étape *« Imaginer »* pour apporter les modifications qui s'imposent. Ces modifications sont ensuite intégrées à un prototype lors d'une nouvelle étape *« Créer »*, ce qui permet d'évaluer à nouveau le jeu. Ce cycle itératif se poursuit jusqu'à ce que l'évaluation du Serious Game corresponde aux attentes.

Cette « série d'étapes » du processus de conception peut être étayée par divers outils. Nous avons déjà passé en revue de nombreux outils théoriques *(p.59 et p.86)* qui pourront s'avérer utiles à tout concepteur de Serious Games. Malgré la variété d'outils disponibles, il peut parfois s'avérer nécessaire d'inventer d'autres outils plus adaptés à un projet donné. Ainsi, nous avons utilisé un outil théorique simple sous forme de tableau lors de la création de *geDriver* *(p.87)*. Mais la création de Serious Games ne se limite pas à des considérations théoriques, ainsi que le fait ressortir la méthodologie que nous venons de proposer. Que ce soit pour la réalisation de prototypes ou pour celle d'un Serious Game finalisé, de nombreux outils techniques permettent de transformer des concepts « sur papier » en jeux vidéo reposant sur un programme informatique. La partie suivante de cette thèse est donc consacrée à l'étude de ces « outils techniques ».

BILAN DE LA PARTIE II : CONSIDERATIONS THEORIQUES SUR LE SERIOUS GAME DESIGN

Afin de synthétiser nos réflexions sur les considérations théoriques relatives aux Serious Game Design, nous pouvons essayer d'analyser la manière dont elles répondent aux deux questions induites par notre problématique *(p.14)* :

1 QUELLES SONT LES SPECIFICITES THEORIQUES DE LA CREATION DE SERIOUS GAMES ?

En comparant les méthodologies de conception de jeux vidéo de divertissement et celles destinées aux Serious Games, nous constatons tout d'abord qu'il existe plusieurs approches de formalisation des processus de création. Au vu des différences entre ces nombreuses approches, il ne semble pas possible de définir une « série d'étapes » universelle pour la conception de jeux vidéo ou de Serious Games. Nous avons néanmoins proposé un modèle composé d'étapes génériques qui semble permettre d'identifier une base commune au sein des différents modèles du processus de conception. Concernant la création de jeu vidéo, il s'agit du *modèle générique ICE* *(p.65)*, composé de trois étapes : *« Imaginer »*, *« Créer »* et *« Evaluer »*. Pour la création de Serious Games, il s'agit du *modèle générique DICE* *(p.103)*, composé de quatre étapes : *« Définir »*, *« Imaginer »*, *« Créer »* et *« Evaluer »*. Ce travail nous permet déjà de constater que le processus de conception de Serious Game se distingue de celui du jeu vidéo par la **présence d'une étape supplémentaire** : la définition du contenu sérieux à transmettre.

Au-delà de cette différence d'ordre général, nos retours d'expériences dans le cadre du *CIFRE* ont permis d'identifier de nombreuses spécificités de la création de Serious Games au sein des autres « étapes génériques » *(p.106)* :

1.1 SPECIFICITES DE LA CREATION DE SERIOUS GAMES POUR L'ETAPE « IMAGINER »

Lors de cette étape, la conception de Serious Game se distingue principalement de celle de jeu vidéo par le fait d'avoir à associer les dimensions « sérieuse » et « ludique » *(p.106)*. Nous avons identifié deux principales approches pour associer ces deux dimensions :

- *L'approche « extrinsèque »*. Inspirée par les théories béhavioristes, cette approche vise à utiliser les deux dimensions de manière séparée au sein du Serious Game.

- *L'approche « intrinsèque »*. Associée aux théories cognitivistes, cette approche consiste à mélanger les deux dimensions de manière à ce qu'il ne soit plus possible de les distinguer.

Le fait de s'appuyer sur l'une ou l'autre de ces deux approches change considérablement le travail de conception. En accord avec les principes de la « vague actuelle » de Serious Games, nous avons particulièrement exploré l'approche « intrinsèque » *(p.108)*. Nous constatons alors

qu'il semble exister deux principaux « vecteurs » permettant de diffuser un « contenu sérieux » pour un Serious Game de type « intrinsèque » *(p.142)* :

- *Par le « contenu du jeu »*. Le contenu sérieux est présenté de manière explicite à l'intérieur du jeu. Le principe de jeu est alors pensé pour que le joueur s'y intéresse.

- *Par les « règles du jeu »*. Le contenu sérieux est intégré au sein des mécanismes de jeu. Le joueur doit alors analyser les règles du jeu afin d'identifier le contenu sérieux qu'il véhicule.

Au-delà de ces différences fondamentales, l'étape *« Imaginer »* de la création d'un Serious Game se rattache à celle d'un jeu vidéo sur certains points. Par exemple, le choix entre les approches *« gameplay expérimental »* et *« jeu de genre » (p.140)* se retrouve également pour la création d'un jeu vidéo de divertissement. Dans un autre registre, la mise en place de systèmes de « tutoriaux » permettant d'expliquer le fonctionnement du jeu n'est pas spécifique au Serious Game *(p.147)*. Bien que cette approche ne soit pas aussi importante pour l'ensemble des jeux vidéo de divertissement, elle est primordiale pour un de ses courants : le *« Casual Game »*. De par la nécessité d'expliquer de manière exhaustive les codes de la culture vidéoludique afin de pouvoir s'adresser à des joueurs novices, la conception de Serious Games rejoint donc celle de *« Casual Games »* sur certains aspects.

1.2 SPECIFICITES DE LA CREATION DE SERIOUS GAMES POUR L'ETAPE « CREER »

La création de prototypes, que ce soit pour un projet de Serious Game ou un jeu vidéo de divertissement, ne s'appuie pas sur des outils théoriques mais sur des outils techniques *(p.44)*. Nous n'avons donc pas étudié les éventuelles différences entre la création d'un jeu vidéo et d'un Serious Game pour l'étape *« Créer »* dans cette deuxième partie, car nous les abordons dans la troisième partie de cette thèse, consacrée aux outils techniques *(p.155)*.

1.3 SPECIFICITES DE LA CREATION DE SERIOUS GAMES POUR L'ETAPE « EVALUER »

Si les jeux de divertissement doivent être évalués tout comme les Serious Games, ils se différencient par les critères qui seront utilisés. Ainsi, le principal critère d'évaluation d'un jeu de divertissement sera la qualité du « fun » qu'il procure au joueur. Si les Serious Games se doivent d'être amusants, ils doivent également être efficaces dans la transmission de leur contenu sérieux. Les critères d'évaluation d'un Serious Game porteront donc à la fois sur ses dimensions « sérieuse » et « ludique ».

Plusieurs méthodes existent pour évaluer un Serious Game. Dans le cas d'un jeu basé sur une approche « intrinsèque », nous avons par exemple pu observer l'intérêt des méthodes de « suivi du joueur » pour évaluer l'impact d'un Serious Game *(p.145)*. Si de telles méthodes peuvent également être mobilisées par le jeu vidéo de divertissement, elles semblent être plus volontairement associées au Serious Game. Nous avons d'ailleurs nous-mêmes expérimentés cette méthode dans deux Serious Games réalisés dans le cadre du *CIFRE*, grâce à des outils de mesure génériques encore peu utilisés pour les Serious Games *(p.147)*. Conjuguée à une analyse qualitative *(p.129)*, l'analyse quantitative des données obtenues par l'intégration de fonctionnalités de suivi du joueur au sein de ces deux Serious Games s'est avérée très pertinente pour l'évaluation de l'impact de nos Serious Games *(p.134)*.

2 QUELLES SONT LES APPROCHES PERMETTANT DE FACILITER LA CREATION DE SERIOUS GAMES ?

S'il n'est pas possible de formaliser de manière universelle le processus de création d'un Serious Game, à travers le *modèle générique DICE* nous proposons un outil théorique permettant à chacun de définir simplement les différentes étapes qu'il utilisera pour créer un tel jeu *(p.103)*. Nous pensons également qu'identifier les spécificités de la conception de Serious Games, comme par exemple les notions d'approches « extrinsèque » et « intrinsèque », pourra aider un novice dans sa réflexion créative. Afin de proposer une approche de facilitation théorique concrète, nous avons synthétisé nos réflexions sur les spécificités de la création de Serious Games à travers une méthodologie détaillée de conception *(p.150)*.

Pour les autres types d'outils théoriques permettant d'accompagner le concepteur durant chacune des phases de son travail, il reste encore beaucoup à faire. En effet, bien que nous ayons identifié de nombreux outils théoriques dans le champ du *« Game Design »*, ceux-ci semble encore relativement peu employés par les concepteurs. Nous avons essayé d'appliquer certains de ces outils pour la création d'un Serious Game, mais ils se sont révélés trop complexes pour les « novices » impliqués dans le projet *(p.87)*. Si ces outils théoriques sont potentiellement intéressants, il semble qu'il faille encore les améliorer pour qu'ils s'avèrent pertinents dans le champ du Serious Game. Nous constatons d'ailleurs que l'invention de nouveaux outils théoriques adaptés aux Serious Games est le sujet de plusieurs travaux de recherche actuels, à l'image du *Game Object Model* *(p.101)*, des *Serious Game Design Patterns* *(p.101)* ou encore des *« widgets »* de ***Marfisi-Schottman & al.*** *(p.91)*.

3 SYNTHESE

En conclusion, au-delà des modèles du processus de conception, nous ne recensons pour l'instant qu'un nombre relativement faible d'approches destinées à faciliter la création de Serious Games. Bien que le travail théorique soit une part très importante de cette création, il n'est pas forcément celui qui nécessite le plus de compétences techniques. La création d'un « bon jeu » est un véritable savoir professionnel, qui s'acquiert avec de nombreuses années d'expérience. Mais un concepteur novice pourra toujours, par exemple, s'inspirer de jeux existants pour « faciliter » cette partie. Cependant, s'il ne possède pas de compétences techniques lui permettant d'implémenter son concept de jeu au sein d'un programme fonctionnel, il ne sera pas en mesure de créer un Serious Game. Si nous avons pu identifier de nombreuses spécificités de la conception de Serious Game sur l'aspect théorique, nous émettons donc l'hypothèse que les approches permettant de faciliter sa création se focalisent plutôt sur l'aspect technique. La vérification de cette hypothèse constitue justement le thème de la troisième partie de cette thèse.

[PARTIE III]
CONSIDERATIONS TECHNIQUES SUR LE SERIOUS GAME DESIGN

Cette partie est consacrée à l'analyse de l'aspect technique de la création de Serious Games, et plus particulièrement aux outils utilisés durant l'étape *« Créer »* du *modèle générique DICE* *(p.103)*. Pour cela, nous analyserons une large sélection « d'outils techniques ». Présentant des niveaux d'accessibilité différents, ces outils peuvent s'adresser aux concepteurs professionnels comme aux amateurs *(p.46)*. En regard de la problématique de cette thèse *(p.14)*, nous avons choisi de nous focaliser sur les « outils techniques » destinés aux amateurs. Ces derniers proposent le niveau d'accessibilité le plus élevé, et donc un potentiel d'utilisation par un public plus large *(p.57)*.

Plus particulièrement, nous nous focaliserons sur les outils de type *« usines à jeux »*, qui permettent de créer de nouveaux jeux vidéo autonomes *(p.166)*. Cela ne nous empêchera pas pour autant d'évoquer le potentiel des outils de *« modding »* pour les Serious Games *(p.163)*. Nous analyserons également des outils techniques faisant partie d'un courant appelé « Jeu 2.0 » *(p.200)*. Nous terminerons notre étude des outils techniques par les logiciels explicitement présentés comme permettant de créer des Serious Games *(p.214)*.

L'analyse de ce large corpus d'outils techniques divers et variés nous permettra d'identifier plusieurs approches de facilitation de l'aspect technique de la création vidéoludique. Issues du *« Game Design »*, ces approches de facilitation semblent applicables à la création de Serious Games *(p.230)*.

METHODOLOGIE D'ANALYSE DES OUTILS TECHNIQUES

Afin d'analyser et présenter les outils techniques de création vidéoludique, nous avons opté pour une méthodologie très simple. Nous avons tout d'abord constitué un large corpus d'outils techniques, et les avons classifiées selon notre propre système : le *modèle ISICO étendu* *(p.160)*. Ce chapitre présente de manière détaillée le cadre théorique et notre méthodologie d'analyse de ce corpus. Nous présentons ensuite une sélection d'usines à jeux ayant joués un rôle historique important, que ce soit par une innovation technique ou par un succès populaire. Cette approche qualitative vise à illustrer les différences d'approches et de fonctionnalités du champ des *« usines à jeux »*. A partir de ce panel, nous espérons identifier des outils techniques applicables au champ du Serious Game *(p.166)*. Nous étudierons ensuite de manière plus précise le courant du « Jeu 2.0 », qui représente un sous-ensemble d'outils aux caractéristiques particulières *(p.200)*. Enfin, nous comparerons tous ces outils issus du *« Game Design »* aux quelques outils techniques explicitement conçus pour le *« Serious Game Design »* *(p.214)*.

1 CADRE THEORIQUE

A l'inverse des « outils théoriques » de « conception » de jeu, qui impliquent de considérer un jeu sous les trois angles définitoires introduits par ***Juul*** *(p.43)*, les « outils techniques » sont principalement destinés à permettre leur « fabrication ». Il ne sont donc a priori concernés que par son premier angle définitoire : **le jeu en tant qu'objet**.

1.1 LE JEU VIDEO EN TANT QU'OBJET

Pour autant, le concept de « jeu en tant qu'objet » reste une notion relativement vague. Afin de pouvoir analyser des « outils techniques » permettant de fabriquer ces « objets jeu », il convient d'avoir de plus amples informations sur la nature de cet objet. En d'autres termes, **est-il possible de formaliser une définition du jeu en tant qu'objet ?**

Différentes approches semblent possible, à l'image de la variété des modèles formels du jeu étudié dans le chapitre précédent *(p.68)*. Aussi intéressant soient-ils, tous ces modèles se révèlent trop détaillés pour permettre une classification générale des « outils techniques ». En effet, pour analyser un corpus d'usines à jeux, nous souhaitons être en mesure de classifier les différents logiciels selon les parties d'un « objet jeu » qu'ils permettent de fabriquer. Et pour cela, nous avons besoin d'un modèle définissant ces différentes « parties ».

1.2 DE LA NATURE D'UN « OBJET JEU »

D'après Crawford (2003), l'interactivité peut être définie comme :

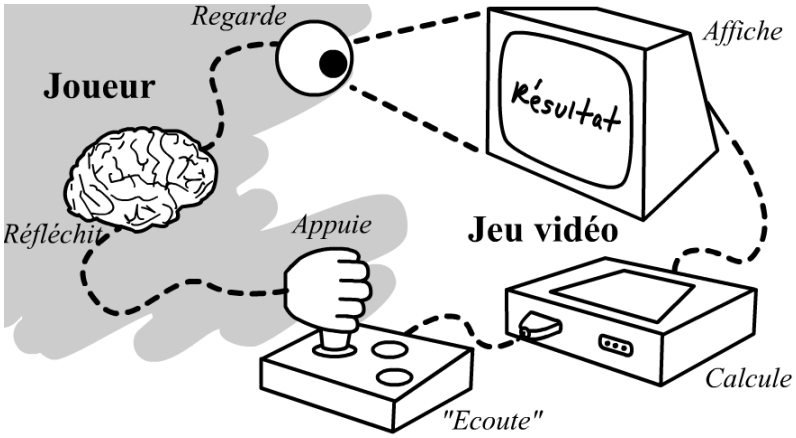
> *« Un processus cyclique dans lequel deux agents (métaphoriques) écoutent, réfléchissent et parlent de manière alternée »*[40].

[40] *"A cyclic process in which two active agents alternately (and metaphorically) listen, think, and speak."*

Si nous considérons qu'un « objet jeu » de type « jeu vidéo » est un de ces deux « agents », et qu'il est en train d'interagir avec un joueur, nous pouvons avancer les associations suivantes :

- *« Ecoute » :* afin de pouvoir écouter un joueur, un jeu vidéo s'appuiera sur son interface entrante, dont la configuration a été imaginée par le concepteur du jeu.
- *« Réfléchi » :* le jeu vidéo va utiliser les règles de jeu inventées par son concepteur afin de réagir aux actions du joueur sur l'interface entrante. Un jeu n'étant a priori pas doté de conscience propre, il semble plus approprié de dire que le jeu « calcule » un résultat. Il se base alors sur les informations provenant de l'interface entrante pour simuler la phase de réflexion.
- *« Parle » :* le résultat de la phase de calcul est communiqué au joueur par le biais d'une interface sortante. En créant des visuels, du son ou encore des séquences de retour de force, le concepteur du jeu décide comment l'objet jeu vidéo utilise son interface sortante afin de communiquer un résultat au joueur.

A ce stade, nous observons donc trois « parties » d'un objet jeu vidéo : une **interface entrante** écoutant les actions du joueur, des **règles** permettant de calculer une réponse appropriée, et une **interface sortante** permettant de la communiquer au joueur.



49. Illustration de la définition de *Crawford* pour un jeu vidéo pratiqué par un seul joueur

Mais ce ne sont pas les seules parties qui semblent composer un « objet jeu ». Lorsque nous avons étudié les définitions du jeu, nous avons observé que ***Juul*** (2003) définissait le jeu en tant qu'objet de la manière suivante :

> *« Un jeu est un système formel basé sur des règles donnant lieu à un résultat quantifiable variable. »*[41]

Dans la suite de ses travaux, ***Juul*** (2005), affine sa vision de « l'objet jeu »:

> *« Au sens littéral, un jeu est une machine à état variable : un jeu est une machine qui peut être dans différents états, qui peut répondre de façon différente à la même entrée, qui possède des fonctions d'entrée et de sortie, ainsi que des définitions liant chacun de ses états à des données entrantes. […] Quand vous jouez à un jeu, vous interagissez avec la machine à état variable qu'est le jeu. »*[42] *[p.60]*

[41] *"A game is a rule-based formal system with a variable and quantifiable outcome"*

[42] "*In a literal sense, a game is a state machine: A game is a machine that can be in different states, it responds differently to the same input at different times, it contains input and output functions and definitions of what state and what input will lead to what following state.[…] When you play a game, you are interacting with the state machine that is the game."*

Cette notion d'un jeu en tant que machine à état variable se retrouve également chez ***Dormans*** (2009) , qui s'appuie sur ce postulat pour proposer un système de représentation formel du jeu par des diagrammes *(p.70)* :

> *« Un jeu peut être vu comme une machine à état variable : il y a un état initial et les actions du joueur (ainsi que celles du jeu) amènent de nouveaux états jusqu'à ce qu'un état final soit atteint. Dans la plupart des jeux pour un seul joueur, soit le joueur gagne, soit le jeu se termine prématurément. L'état courant d'un jeu reflète généralement la position du joueur, la position des autres joueurs, des alliés, des ennemis, et la répartition actuelle des ressources du jeu. A partir de l'état courant d'un jeu, il est possible d'analyser la progression du joueur vers l'objectif qu'il doit atteindre pour gagner. »*[43]

La vision de ***Dormans*** a été influencée par le travail de ***Grünvogel*** (2005), qui voit les jeux comme un ensemble d'objets à état variables :

> *« En faisant temporairement abstraction des dimensions sociale, psychologique et culturelle, un jeu est composé d'objets dont l'état change durant la partie. Les changements d'états de chaque objet sont régit par les règles du jeu et influencés par les joueurs ou par les autres objets. »*[44]

En résumé, ces différentes approches nous incitent à considérer le jeu comme une machine à état variable. Les états de cette machine sont eux-mêmes le fruit des changements d'états d'objets internes. Nous pouvons alors voir un jeu non pas seulement comme une machine à état variable, mais bel et bien comme **un système à état variable**, en tout cas au sens où l'entendent ***Salen & Zimmerman*** (2003) :

> *« Un système est un ensemble d'éléments mis en relation de manière à former un tout plus complexe »*[45] *[p.55]*

Ce système a état variable se caractérise par la présence d'un état initial, autrement dit l'état du jeu lorsque la partie commence. Nous avons vu qu'il existe de nombreux outils permettant de créer des « niveaux », des « circuits » et autres « cartes » *(p.50)*. Concrètement, ces « éditeurs de niveaux » permettent au créateur de disposer un ensemble d'objets pour créer un espace de jeu, puis de définir l'état de départ de chacun de ces objets. Par exemple, pour un jeu de plateforme, l'éditeur permettra de placer les différents blocs sur lesquels l'avatar du joueur pourra marcher, mais aussi de disposer des bonus à ramasser et des ennemis. Pour les ennemis, il pourra être proposé de définir certains paramètres, comme le nombre de points de vie au départ ou l'arme à utiliser pour attaquer le joueur. A la lumière des définitions exposées précédemment, de tels « éditeurs de niveaux » permettent tout simplement de **créer l'état initial du système à état variable** qu'est « l'objet jeu ». Nous avons déjà mis en évidence trois « parties » d'un jeu tant qu'objet : *l'interface entrante*, *les règles* et *l'interface sortante*. Nous en identifions ici une quatrième : **l'état initial**.

Rappelons que la création de l'état initial d'un jeu, bien que faisant partie du processus de *« Game Design »*, est généralement désignée par l'expression *« Level Design » (p.45)*.

---

[43] *"A game can be understood as a state machine: there is an initial state or condition and actions of the player (and often the game, too) can bring about new states until an end state is reached. In the case of many single-player video games either the player wins or the game ends prematurely. The game's state usually reflects the player's location, the location of other players, allies and enemies, and the current distribution of vital game resources. From a game's state the player progress towards a goal can be read"*

[44] *"Leaving aside for a moment the social, psychological and cultural aspects of a game, a game consist of objects which change their state during the play, where the evolution of their state is governed by the rules and influenced by the players or other objects."*

[45] *"A system is a set of parts that interrelate to form a complex whole"*

1.3 LE MODELE ISICO

En conclusion, nous avons donc identifié quatre « parties » qui composent un jeu, la notion de « jeu » étant ici entendue comme un « objet » créé par un concepteur :

- **Initial State** : l'état initial du système à état variable qu'est le jeu.
- **Input** : les moyens proposés au joueur pour envoyer des informations au système.
- **Compute** : les mécanismes internes qui permettent au système de changer d'état.
- **Output** : la manière dont le système communique son état courant au joueur.

Ces quatre parties sont les différentes composantes créées par un concepteur lorsqu'il conçoit un « objet jeu ». Nous proposons d'appeler ce modèle théorique le *modèle ISICO (abréviation de : **I**nitial **S**tate, **I**nput, **C**ompute, **O**utput)*. Ce modèle se focalise uniquement sur les différentes composantes d'un jeu en tant qu'objet. Il n'est donc pas destiné à l'analyse des jeux, mais plutôt à l'analyse des « outils techniques » permettant de créer des jeux. En utilisant ce modèle comme grille de lecture, il est possible de classifier les outils techniques de création vidéoludique selon les « parties » d'un jeu vidéo qu'ils permettent de créer.

2 ANALYSE D'UN LARGE CORPUS D'OUTILS TECHNIQUES

L'objectif visé par ce travail est de présenter une sélection d'outils techniques permettant de créer des jeux vidéo, et de les classifier par un système adapté. Les résultats attendus de cette approche exploratoire sont l'identification d'approches de facilitation de la création de jeux vidéo. Ces éventuelles approches seront alors évaluées pour la création de Serious Games.

2.1 METHODOLOGIE

A ce jour, il n'existe pas à notre connaissance de liste exhaustive d'outils techniques de création vidéoludique destinés aux amateurs. Par contre, plusieurs listes de références plus ou moins complètes existent sur Internet. Par exemple, sur le site *Ambrosine* nous trouvons une liste de 125 logiciels de création vidéoludique [46], sur *Wikipedia* une liste de 133 moteurs de jeu[47], sur *If-Wiki* sont référencés 76 outils dédiés à la fiction interactive[48]…

2.1.1 CLASSIFICATION DE 380 OUTILS TECHNIQUES AVEC LE MODELE ISICO

Nous avons commencé par explorer et analyser ces diverses listes, que nous avons complétées par de nombreuses références supplémentaires obtenues par le biais de recherches empiriques. Tel qu'exposé précédemment, notre attention s'est portée sur les « usines à jeux », même si nous avons également référencés d'autres types d'outils *(SDK, éditeur de niveaux, Jeu 2.0…)*, afin d'éprouver la fiabilité de notre système de classification. Un corpus d'exactement **380 outils techniques** a ainsi été rassemblé, puis classifié selon le *modèle ISICO*. Pour classifier un outil, nous avons tout simplement regardé s'il permettait de créer et/ou modifier chacune des quatre « parties » définies par le *modèle ISICO*.

D'un point de vue méthodologique, nous avons été contraints d'effectuer des choix sur le nombre d'entrées dans le corpus attribuées à chaque outil. En effet, comme de nombreux logiciels destinés à la création, les outils techniques connaissent une évolution constante. Pour les logiciels distribués par Internet, il est très courant de voir des mises à jour mensuelles, voire quotidiennes pour les logiciels open-source. Il est bien évident que nous n'avons pu référencer chacune de ces versions au sein de notre corpus. Nous avons donc donné priorité

[46] Relevé le 23-02-11 sur http://www.ambrosine.com/resource.php
[47] Relevé le 23-02-11 sur http://en.wikipedia.org/wiki/Category:Video_game_engines
[48] Relevé le 23-02-11 sur http://www.ifwiki.org/index.php/Category:Authoring_system

aux versions « majeures » des logiciels *(1.0, 2.0, 3.0…)*, ou à défaut aux versions qui introduisent des améliorations modifiant la classification de l'outil *(passage en open-source, ajout d'un nouveau langage de programmation…)*. L'objectif d'un tel choix est de récolter des données historiques sur le champ des usines à jeux, sans pour autant biaiser le corpus en « sur-référençant » les outils qui connaissent des mises à jour régulières.

Le *modèle ISICO*, prometteur lorsque seule une dizaine de logiciels étaient classifiés, a montré ses limites une fois appliqué à l'intégralité du corpus de 380 outils techniques. Empiriquement, de nombreuses informations semblaient manquer lors de la lecture du corpus. Par exemple, face à un groupe de 307 outils permettant de créer/modifier la partie *« Compute »* d'un jeu vidéo, il nous manquait des informations plus précises sur les modalités de création : *utilisation ou non d'un langage de programmation, présence ou non d'un éditeur visuel…* De même, le fait de savoir si un outil permet de créer de nouveaux jeux autonomes *(usine à jeux)* ou s'il permet uniquement de modifier un jeu existant *(outil de modding)* semble primordial. Cette information est pourtant absente du *modèle ISICO*.

2.1.2 LE MODELE ISICO ETENDU

A partir de l'analyse des caractéristiques de ce premier corpus de 380 outils techniques, nous avons donc entrepris d'étendre le *modèle ISICO* en lui rajoutant des « sous-catégories » qui permettrait d'affiner la classification de chacune de ses quatre parties. Nous avons également rajouté deux critères de premier niveau, destinés à renseigner plus précisément sur le type du logiciel. En premier lieu, chaque logiciel se verra dorénavant classé selon qu'il permette de *« créer des nouveaux jeux autonomes »* ou de *« modifier un jeu existant »*. Nous avons également rajouté un critère pour noter la présence d'une éventuelle *« plateforme de partage intégrée »*, comme c'est le cas des outils de la famille du « Jeu 2.0 » *(p.200)*. L'ajout de ces nouveaux critères et de sous-catégories pour chacune des quatre parties du *modèle ISICO* permet de définir le *modèle ISICO étendu*. Ce modèle est composé des critères suivants :

- **Type :**
  - Création de nouveaux jeux autonomes OU Modification d'un jeu existant
  - Présence d'une plateforme d'échange intégrée
- **Initial State** *(Etat Initial)* :
  - Importation (fichier texte...)
  - Sélection dans une librairie
  - Editeur visuel
- **Input** *(Interface entrante)* :
  - Clavier
  - Souris
  - Divers (joystick, manette, tapis de danse…)
- **Compute** *(Règles)* :
  - Configuration de paramètres
  - Editeur visuel
  - Langage de script propriétaire
  - Langage de programmation commun
- **Output** *(Interface sortante)* :
  - Importation (image, son…)
  - Sélection dans une libraire
  - Editeur visuel
  - Représentation 2D
  - Représentation 3D
  - Représentation textuelle

A une exception près, ces critères sont tous de type « case à cocher » : pour classifier un outil, il suffit de noter si oui ou non il propose chaque caractéristique listée par le modèle.

Si nous avons déjà explicité les critères du *« Type »*, nous pouvons faire de même pour les « sous-catégories » rajoutées aux quatre parties du *modèle ISICO*. *« Importation »* désigne le fait que l'outil ne propose pas de moyen direct pour créer ou modifier la partie concernée, mais permet de la modifier grâce à des fichiers générés par un autre logiciel. Par exemple, l'usine à jeux de combats *M.U.G.E.N.* ***(Elecbyte, 1999-2011)*** ne propose pas de moyen pour éditer l'état initial, mais n'importe quel éditeur de texte permettra de créer des fichiers que l'outil interprétera à cette fin. D'une manière plus générale, la plupart des usines à jeux permettent au créateur d'utiliser des fichiers images ou sons qui auront été créés à l'extérieur de l'outil. Le critère *« Sélection dans une libraire »* désigne quant à lui le fait que l'outil propose une liste de choix limités pour la partie concernée. Par exemple, le logiciel *The Sims Carnival Wizard* ***(Electronic Arts, 2008)*** propose de créer l'état initial du jeu uniquement en choisissant entre trois ou quatre « niveaux » préconstruits. Si le créateur souhaite pouvoir lui-même construire son niveau à partir d'éléments de base, il devra se tourner vers un logiciel classifié avec le critère *« Editeur visuel »* pour la partie *« Etat Initial »*. Cet « éditeur visuel » permet de créer du contenu qui n'a pas forcément été anticipé par le créateur de l'outil. A l'inverse, d'autres outils proposent des menus qui ne permettent de régler qu'une liste de paramètres présélectionnés. Ils sont alors désigné par le critère *« Configuration de paramètres »*. Loin d'être anecdotique, cette différence est fondamentale, et renvoie au fait qu'il existe **deux grandes philosophies** d'éditeurs au sein des logiciels de création vidéoludique que nous avons étudiés :

- **Les menus de choix prédéfinis**. Ils proposent aux créateurs de jeux une série de choix déterminés par le concepteur du logiciel. Bien que très simples à utiliser, ces outils restreignent la créativité aux limites de l'imagination du concepteur de l'outil. *Exemples : menus d'options permettant de configurer les règles du jeu, bibliothèques d'éléments graphiques à sélectionner…*

- **Les éditeurs de contenu émergents.** Il s'agit d'un ensemble de fonctionnalités simples qui permettent de créer du contenu « émergent », autrement dit un contenu qui n'aura pas été forcément anticipé par le concepteur de l'outil. Bien que plus complexes à prendre en main, ils offrent une liberté de création nettement supérieure aux menus de choix prédéfinis.
  Exemples : éditeurs de dessins, langages de script, éditeurs de niveaux…

Au delà de cette distinction fondamentale, nous avons également différencié les outils ayant recours à un langage de programmation pour la partie *« Compute »*. Le *« langage de script propriétaire »* désigne un langage de programmation inventé pour le logiciel, et qui lui est généralement spécifique. Ces types de langages s'opposent aux *« langages de programmation commun »* tel que *BASIC*, *Java*, *C++* ou *.NET*, dont le champ d'application dépasse celui d'une gamme de logiciels unique. Les langages de scripts sont néanmoins plus simples à apprendre qu'un langage de programmation commun pour un novice. Ils s'inspirent d'ailleurs très largement des langages de programmation communs qu'ils cherchent généralement à simplifier. Toujours pour la partie *« Compute »*, le critère *« Editeur visuel »* désigne les logiciels dont la programmation se fait sans passer par un quelconque langage de programmation. Dans un autre registre, nous avons également rajouté des critères sur l'aspect *« Input »*. La configuration de l'interface entrante étant généralement laissée à la discrétion du joueur par le biais de menus d'options *(p.53)*, nous nous sommes contentés de noter si le logiciel permettait de créer des jeux utilisant le clavier, la souris, ou d'autres types de périphériques de contrôle. Enfin, pour l'aspect *« Output »*, nous classifions les outils selon

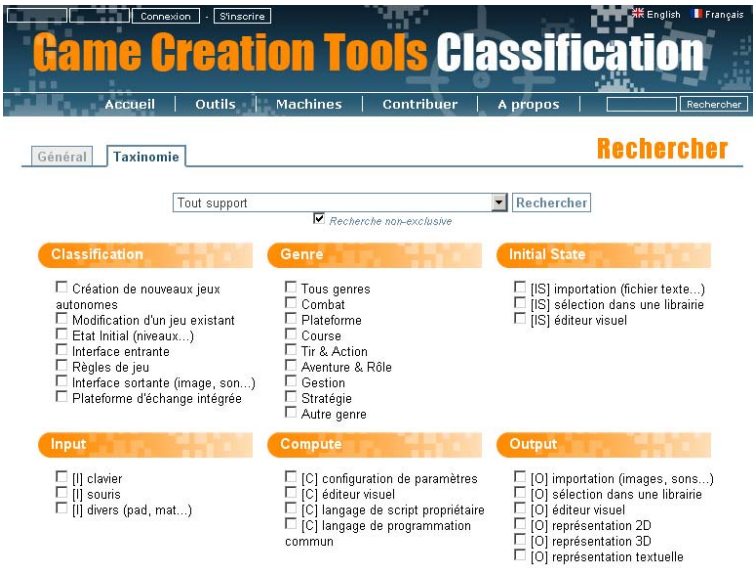
qu'ils permettent de créer des jeux s'appuyant sur une représentation textuelle, de graphisme 2D, ou de 3D temps réel.

2.1.3 APPLICATION DU MODELE ISICO ETENDU A UN CORPUS DE 400 OUTILS

Nous avons alors utilisé ce *modèle ISICO étendu* sur un corpus total de **400 outils techniques**. Ce nouveau corpus reprend la liste des 380 outils de notre premier corpus appliqué au modèle *ISICO*, étoffée de 20 références supplémentaires. La classification de ce corpus de 400 outils a été reprise à zéro afin de pouvoir appliquer correctement le *modèle ISICO étendu*. Précisons que lors de la constitution de ce corpus, nous avons mis l'accent sur les outils techniques de type *« usines à jeux »*, en accord avec notre problématique *(p.57)*. Afin d'éprouver notre modèle classificatoire, nous y avons cependant intégré quelques outils de « modding » *(p.50)*. Nous étudierons brièvement leur potentiel pour le Serious Game dans la section suivante. Nous analyserons ensuite de manière beaucoup plus détaillée les nombreuses *« usines à jeux »* de notre corpus *(p.166)*, ainsi que des représentants du *« Jeu 2.0 »* *(p.200)* et quelques outils explicitement destinés aux Serious Games *(p.214)*.

2.1.4 CONSTRUCTION D'UNE BASE DE DONNEES EN LIGNE DES OUTILS TECHNIQUES

Afin de faciliter l'accès à notre corpus, nous avons créé un site Internet qui rassemble les 400 outils techniques que nous avons étudiés au sein d'une base de données. Ce site permet tout d'abord de consulter des informations basiques sur les outils : *nom, année de publication, nom et pays d'origine du développeur et de l'éditeur, mode de diffusion…*Cette base de donnée recense également les informations de classification selon *le modèle ISICO étendu*. Il est donc possible d'effectuer des recherches d'outils techniques en utilisant les différents critères de ce modèle, tel qu'illustré par la capture d'écran ci-dessous :



50. Recherche d'un outil technique grâce au *modèle ISICO étendu* sur notre site Internet

Sur un modèle similaire au site *Serious Game Classification* *(p.32)*, ce site est également ouvert aux contributions extérieures selon les modalités collaboratives du *« Web 2.0 »*. Ce site Internet détaillant notre corpus est accessible ici : http://creatools.gameclassification.com

Au sein de notre corpus de 400 outils techniques, **37 outils de « modding »** ont été identifiés *(critère « modification d'un jeu existant »)*. Cette petite sélection nous invite à étudier le potentiel apport de la pratique du *« modding »* *(p.51)* pour les Serious Games.

2.2.1 LE RECOURS AU « MODDING » POUR LA CREATION DE SERIOUS GAMES

Nous remarquons tout d'abord que certains créateurs de Serious Games choisissent de s'appuyer sur le « modding » pour transmettre leur message. Par exemple, *Escape from Woomera* ***(Kate Wild et al., 2003)*** modifie le jeu *Half-Life* ***(Valve Software, 1998)*** pour alerter l'opinion publique sur les conditions de vie dans les camps de réfugiés situés en Australie *(p.25)*. Dans un autre registre, *Marine Doom* ***(1996)*** est un « mod » à vocation militaire du jeu *Doom II* ***(id Software, 1994)***. Il est créé directement par l'armée américaine, à travers le ***Marine Corps Modeling and Simulation Management Office***, dont le ***Lieutenant Scott Barnett*** endosse le rôle de chef de projet. Ce « mod » est dédié à l'entraînement des troupes. Axé sur les capacités de jeu en réseau de *Doom II*, le « mod » *Marine Doom* permet à des escouades de quatre soldats de s'entraîner à agir ensemble, avec pour mission d'arriver à neutraliser un bunker ennemi. Enfin, *Revolution* ***(The Education Arcade, 2004)***, permet au joueur de vivre une période historique particulièrement importante aux Etats-Unis : celle de la guerre d'indépendance de 1775. Il se présente sous la forme d'un jeu de rôle multijoueurs, dans lequel chacun incarne un personnage d'une classe sociale différente : d'esclave à notable, en passant par un forgeron patriote. Pensé pour l'usage en classe, ce titre permet aux élèves d'appréhender les différentes perspectives du conflit, dont ils pourront ensuite débattre. En effet, ce titre est calibré pour des sessions d'environ 45 minutes, le rendant compatible avec une introduction ou des débats menés par l'enseignant en complément de la pratique du jeu. Ce Serious Game produit par des chercheurs du ***MIT*** est un « mod » du jeu de rôle *Neverwinter Nights* ***(Bioware, 2002)***.

2.2.2 LA CREATION DE « MODS » COMME METHODE PEDAGOGIQUE

Loin de se limiter à la seule création de Serious Games, la production de « mods » peut en elle-même être le berceau d'une activité pédagogique. Par exemple, la création de « mods » est utilisable pour l'enseignement de l'informatique, comme l'ont étudié ***El-Nasr & Smith*** (2006). Leur étude présente deux expérimentations très intéressantes. La première est menée sur une classe de lycéens, lors d'un cours d'initiation à l'informatique. Afin de leur faire approcher des concepts de programmation informatique *(paradigme objet, variables, programmation événementielle…)*, les enseignants proposèrent aux élèves de réaliser un « mod » pour le jeu *Warcraft III* ***(Blizzard Entertainment, 2002)***. Il s'agit d'un jeu de « stratégie temps réel (STR) », qui demande au joueur de gérer des ressources afin de construire une base et une armée, puis de commander cette armée pour détruire l'adversaire. Ce titre est livré avec un *SDK* permettant de modifier la plupart des aspects du jeu. Dans l'expérimentation de ***El-Nasr & Smith***, des lycéens ont transformé ce jeu en simulation de football américain. Ce faisant, ils ont tout d'abord du comprendre les règles du jeu d'origine. Ils ont ensuite fait appel à des concepts de programmation événementielle et de logique Booléenne pour programmer leur « mod ». Dans un registre similaire, un autre groupe de lycéen transforma *Warcraft III* en un jeu de réflexion inspiré par *Tetris* ***(Alexey Pajitnov, 1985)***. Ces élèves furent ainsi amenés à réviser les notions de trigonométrie pour programmer le mouvement et la rotation des objets de leur « mod ». Un des éléments permettant à ces lycéens, sans connaissances particulières en informatique, de programmer de tels jeux est l'emploi des outils proposés par le SDK de *Warcraft III*. Par exemple, « l'éditeur de niveaux » permet aux « moddeurs » de créer un nouveau de terrain de jeu de manière visuelle et intuitive. Cette logique est conservée dans « l'éditeur d'événements », qui permet de créer des instructions de programmation de manière « graphique ». Concrètement, l'utilisateur peut associer des « blocs d'actions » à des « conditions » préprogrammées afin de créer la logique

de son jeu. Il est également possible d'inventer de nouvelles « conditions » et « blocs d'actions » grâce à un langage de script dédié : *JASS*. Ce principe de fonctionnement est directement inspiré des usines à jeux que nous étudierons en détail dans le chapitre suivant *(p.166)*. Au final, les deux chercheurs notent que l'utilisation de ces outils a permis à des lycéens, réunis en petits groupes de travail, de créer leurs jeux en seulement cinq jours, temps de découverte des outils et de formation théorique inclus.

El-Nasr & Smith ont également mis en place une expérimentation similaire avec des étudiants plus âgés, en l'occurrence des étudiants en licence d'informatique. La réalisation de jeu vidéo a ici pour objectif d'enseigner par la pratique les principes de base de la programmation informatique. Pour cela, deux outils furent utilisés. D'un côté l'usine à jeux *WildTangent Web Driver* ***(WildTangent, 1999)***, et de l'autre le jeu *Unreal Tournament 2003* ***(Epic Games, 2002)*** qui propose un langage de script propriétaire baptisé *UnrealScript*. Pour s'adapter à un fonctionnement de module de Travaux Pratiques en milieu universitaire, les étudiants se sont vus proposer de nombreux exercices à réaliser pour chacun des deux outils. Ce faisant, ils mobilisaient leur cours théoriques sur la programmation informatique pour réaliser des « mods » de jeux (avec *Unreal Tournament 2003*) ou de nouveaux jeux autonomes (avec *WildTangent Web Driver*). Les chercheurs ont observés que les exercices de « modding » du jeu *Unreal Tournament 2003* furent nettement plus motivants pour leur groupe de 35 étudiants. Par exemple, chaque module proposait des exercices facultatifs permettant de gagner quelques points bonus. Avec *WildTangent Web Driver*, seul 40% des étudiants ont choisi de réaliser ces exercices facultatifs, alors que 100% de ces mêmes étudiants les ont effectué dans le module *Unreal Tournament 2003*. De plus, un questionnaire de fin de module révèle que 60% des étudiants ont trouvé les Travaux Pratiques avec *Unreal Tournament 2003 « très motivants »*, contre seulement 30% pour ceux dédiés à *WildTangent Web Driver*. Les deux chercheurs imputent ces différences à l'existence d'un jeu de base à modifier pour le premier outil, alors que tout est à créer avec le second. En nous référant aux expériences du groupe ***Pédagame*** (Llanas, 2009), nous pouvons également avancer l'hypothèse que le fait de travailler sur un jeu vidéo commercial à succès augmente la motivation des élèves. Le « modding » leur donne l'impression de « créer un jeu vidéo professionnel » d'une qualité technique comparable à ceux qu'ils utilisent pour se distraire à la maison. Une autre étude vise à explorer l'utilisation du « modding » pour enseigner auprès d'un public féminin (Yucel, Zupko, & El-Nasr, 2006). Après une délicate phase de sélection d'un jeu vidéo qui soit modifiable et qui corresponde aux goûts d'une audience uniquement féminine, le choix des chercheurs s'est porté sur *Warcraft III*. La création de « mod » pour ce jeu visait à permettre l'acquisition d'un nombre déterminé de compétences du domaine *« Technologies de l'Information et de la Communication »*, pour un public de collégiennes et lycéennes. Cette étude montre que le choix d'un titre adapté au public visé, aussi bien en terme de contenu qu'en terme de complexité technique des outils proposés, permet d'impliquer fortement les élèves dans les exercices proposés.

Pour le domaine de l'éducation à l'informatique, ces trois expérimentations mettent en lumière le fort impact de la création vidéoludique sur la motivation des élèves. Leurs conclusions rejoignent donc celles de travaux qui analysent l'utilisation de Serious Games, au lieu d'outils de création vidéoludique, pour ce même domaine (Muratet, 2010).

2.2.3 SYNTHESE : UNE APPLICATION UTILITAIRE DU « MODDING »

Nous constatons tout d'abord que les outils de « modding » permettent bel et bien de créer des Serious Games. Nous avons malheureusement fait le choix de ne pas explorer plus en détail leur potentiel dans ce domaine pour nous concentrer sur les *« usines à jeux » (p.57)*, mais ces quelques exemples illustrent si besoin est que l'étude des outils de « modding » comme approche de facilitation de la création de Serious Games mériterait d'être explorée dans nos futurs travaux.

Nous identifions également un autre apport utilitaire des outils de « modding ». Plusieurs travaux de recherche illustrent qu'ils ont été utilisés avec succès dans un contexte pédagogique. Si, dans ces expérimentations, les outils de « modding » n'ont pas été utilisé pour créer des Serious Games mais des jeux vidéo de divertissement, l'activité même de création de ces jeux à par contre été associée à une finalité « sérieuse » : l'enseignement de l'informatique. S'il s'agit ici uniquement modifier des jeux vidéo existants, nous supposons que les outils permettant de créer de nouveaux jeux autonomes *(usines à jeux…)* possèdent un potentiel similaire. Comme nous le verrons dans la dernière partie de cette thèse, les thèmes abordables au travers de la création vidéoludique dépassent largement le seul cadre de l'enseignement de l'informatique, pour peu que la création de jeux de divertissement soit remplacée par la création de Serious Game *(p.235)*. Mais quel que soit le but visé, il semble que la sélection des outils importe, comme le précisent ***El-Nasr & Smith*** (2006) :

> *« Il ne suffit pas de donner aux étudiants des moteurs de jeux de manière arbitraire pour s'attendre à ce qu'ils apprennent tout seul l'informatique. Nous avons sélectionnés ces trois moteurs de jeux car nous pensions qu'ils pouvaient faciliter l'acquisition de compétences bien précises »*[49]

Ainsi, s'il existe de nombreux outils permettant de s'adonner à la création vidéoludique, leurs différences les rendent plus ou moins pertinents selon les objectifs pédagogiques visés. La connaissance d'une large gamme d'outils et de leurs caractéristiques semble donc primordiale. Cette conclusion motive donc le recensement et la classification des outils techniques de manière approfondie, tels que nous le proposons pour les *« usines à jeux »* dans le chapitre suivant.

[49] *"It is not enough to give students arbitrary game engines and expect them to learn computing skills. We chose the three game engines in our courses because we believed they could assist in the development of particular skills."*

LES USINES A JEUX

Au sein de notre corpus d'outils techniques, **363 usines à jeux** ont été identifiées *(critère « création de nouveaux jeux autonomes »)*. Si nous ne prétendons pas à l'exhaustivité, nous avons précédemment noté que le thème des « usines à jeux » bénéficie d'une littérature très limitée *(p.49)*. Nous proposons alors d'utiliser ce corpus pour proposer une vue d'ensemble de ce champ, tout en étant conscient des limites qu'impose le fait de baser notre analyse sur un corpus de seulement 363 éléments.

Nous commencerons par présenter les « usines à jeux » de manière qualitative. Si la place nous manque pour présenter en détail tous les logiciels présents dans notre corpus, nous pouvons néanmoins en présenter une sélection. Nous avons tout d'abord choisi de mettre l'accent sur les usines à jeux ayant eu une importance historique, et de présenter ces exemples qualitatifs en essayant de respecter une certaine chronologie. Le choix de ces exemples a également été influencé par la volonté d'illustrer la diversité des approches permettant à des amateurs de créer des jeux vidéo. Nous détaillons donc ici les principales approches de facilitation de la création vidéoludique que nous avons pu observer dans notre corpus de 363 usines à jeux. Nous complèterons ensuite ces exemples qualitatifs par des données quantitatives issues de l'ensemble du corpus *(p.190)*.

Avant de présenter les exemples qualitatifs, une distinction primordiale doit être faite entre deux catégories d'usines à jeux : les logiciels **spécialisés dans la création d'un genre de jeu en particuli**er, et les **usines à jeux « généralistes »** qui permettent de créer tout type de jeux vidéo, abstraction faite d'éventuelles limitations techniques. Nous traiterons dans un premier temps les usines à jeux « spécialisées » *(p.166)* avant de nous intéresser à leurs homologues « généralistes » *(p.177)*.

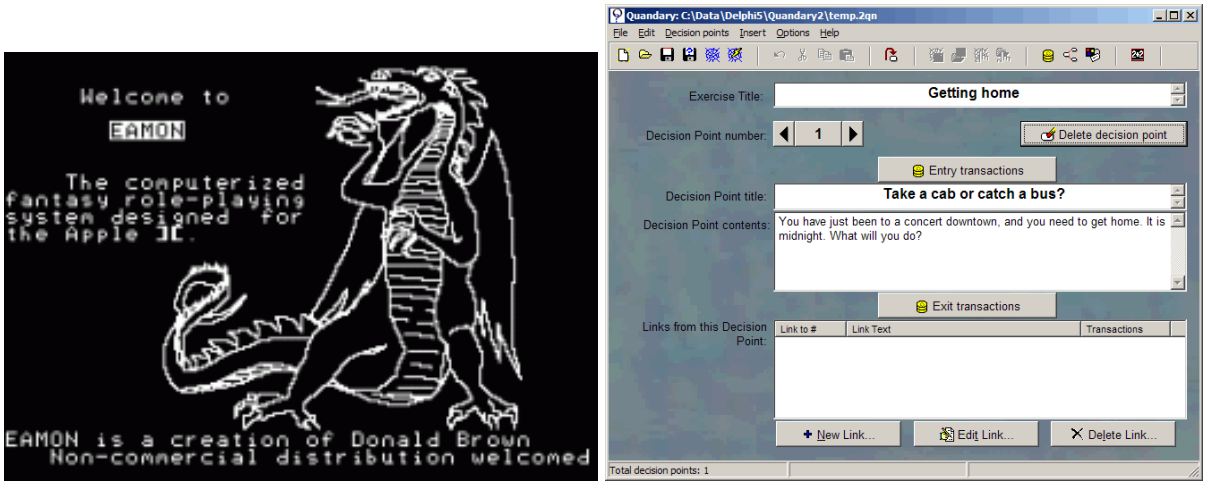
1 LES USINES A JEUX SPECIALISEES

Les logiciels présentés dans cette catégorie permettent à l'utilisateur de créer uniquement des jeux d'un genre donné. La notion de « genre » est en constante évolution (Apperley, 2006; Letourneux, 2006), et il n'existe pas à ce jour de liste de genres vidéoludiques exhaustive et faisant consensus. De manière à néanmoins proposer une classification des usines à jeux spécialisées selon le genre de jeu qu'elles permettent de créer, nous avons utilisé la liste suivante : *Combat, Plateforme, Course, Tir & Action, Aventure & Rôle, Gestion, Stratégie, Autre genre*. Cette liste est issue d'une analyse synthétique personnelle des classifications par genre *(p.28)*.

1.1 DU JEU D'AVENTURE TEXTUEL A LA FICTION INTERACTIVE

L'exemple le plus ancien d'usine à jeux « spécialisée » que nous recensons à ce jour est *Eamon* ***(Donald Brown, 1980)***. Explicitement destiné à une diffusion non commerciale, ce programme est pensé pour la création de jeux d'aventure en mode texte. Pour cela, il propose des éditeurs permettant de modifier le contenu du jeu d'aventure de base, et de redistribuer le tout sous forme d'un nouveau jeu complètement indépendant *(p.49)*. Ce logiciel semble avoir ouvert la voie à de nombreux autres outils destinés à faciliter la création de jeux d'un genre similaire, aujourd'hui connu sous le nom de *« fiction interactive »*. De l'antique *Generic Adventure Game System* ***(Mark Weilch, 1985)*** aux plus récents *Dreampath* ***(Drake Vision***

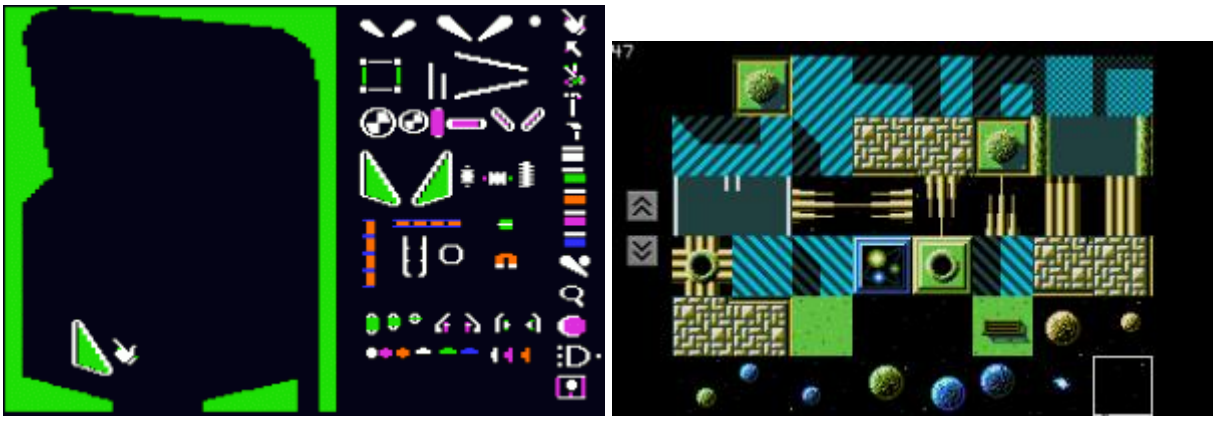
Software, 2008) et *StoryWorld Authoring Tool* ***(Chris Crawford, 2009)***, en passant par les populaires *Inform 7* ***(Graham Nelson, 2006)*** et *TADS3* ***(Michael J. Roberts, 2006)***, 60 outils du genre sont référencés dans notre corpus. S'ils proposent toujours la possibilité de créer des histoires interactive ou des jeux d'aventures en mode texte, ces logiciels ont lentement évolué avec le temps. Les outils les plus récents permettent maintenant d'associer des images fixes au texte. Mais surtout, les éditeurs proposés sont nettement plus intuitifs. Là où les pionniers s'appuyaient sur de simples éditeurs de texte, les outils plus récents proposent des environnements de travail élaborés. Par exemple, une fonctionnalité permet d'afficher les différents « événements » d'une histoire interactive sous forme d'une carte matérialisant les liens qui les unissent. Le concepteur du jeu a ainsi un meilleur aperçu de son travail.



51. *Eamon* (gauche) et *Quandary* (droite)

Bien que ces outils soient à la base pensés pour créer des jeux vidéo destinés au divertissement, il est tout à fait possible de les utiliser à des fins utilitaires. C'est par exemple le cas de *Quandary* ***(Half-Baked Software, 2004)*** qui a été utilisé avec succès en classe par ***Céline Dunoyer*** de l'Académie de Créteil. Elle a intégré ce logiciel au sein d'une activité pédagogique, destinée à des élèves de 6e, visant à leur faire travailler l'écriture, le schéma narratif ainsi que la création en groupe (Dunoyer, 2001). Parmi les nombreux outils disponibles, l'enseignante a opéré un choix en adéquation avec le public visé. Elle a choisi un logiciel en français, simple d'utilisation, et qui permette de générer des histoires diffusables sous forme de site Internet. Le processus de création est très simple : le créateur commence par écrire une description *(par exemple « vous vous trouvez face à une porte »)*. Il peut ensuite proposer plusieurs actions au joueur *(telles que « ouvrir la porte », « faire demi-tour »…)*. Pour chacune de ces actions, il pourra écrire une nouvelle description *(« vous ouvrez la porte, et découvrez un mystérieux chemin »)* et proposer de nouvelles actions. Selon cette logique, il est possible de créer des histoires interactives particulièrement riches, d'autant plus qu'il est également possible d'ajouter des images aux descriptions rédigées. Si ce logiciel est plus limité que les populaires *Inform* ou *TADS*, il est néanmoins plus facile à prendre en main. *Quandary* permet de créer des histoires simples grâce à une interface graphique où il suffit de cliquer sur des boutons et de rédiger des descriptions. A l'inverse, les deux logiciels phares du secteur de la fiction interactive, *Inform* et *TADS*, nécessitent l'apprentissage d'un langage de script propriétaire relativement complexe. Mais ce dernier permet de mettre en place des interactions plus élaborées. Au final, l'existence de plusieurs outils de puissance et complexité d'utilisation variées permet de choisir un logiciel adapté au public que l'on souhaite exposer à la création vidéoludique.

Après *Eamon*, d'autres usines à jeux sont rapidement apparues pour proposer de créer d'autres genres de jeux. Ces jeux s'appuient d'ailleurs sur une représentation graphique et sonore plus évoluée : des graphismes en 2D au lieu du texte. L'outil le plus emblématique en la matière est *Pinball Construction Set* ***(Bill Budge, 1983)***. Comme son nom l'indique, ce logiciel pionnier permet de créer des jeux de flipper. Les divers éditeurs du logiciel permettent de composer une table de flipper en posant divers éléments *(bumpers, trous…)*, puis d'en définir les règles de bases *(nombre de balles…)*. Le jeu est ensuite exportable sous forme autonome, ce qui permet de le diffuser librement. Le succès critique et commercial rencontré par cet outil a inspiré la création de nombreux logiciels similaires, toujours destinés à la création d'un genre vidéoludique en particulier (Barton & Loguidice, 2009). Par exemple, *Adventure Construction Set* ***(Stuart Smith, 1984)*** permet de réaliser des jeux de rôle et d'aventure dans la veine des premiers opus de la série *Ultima* ***(Richard Garriot, 1980-2009)***. De son côté, *Racing Destruction Set* ***(Rick Koenig, 1985)*** permet de créer des jeux de course, alors que *Wargame Construction Set* ***(Strategic Simulations Inc, 1986)*** se focalise sur les jeux de stratégie militaire au tour par tour.



52. ***Pinball Construction Set (gauche)*** **et** ***Shoot'Em-Up Construction Kit (droite)***

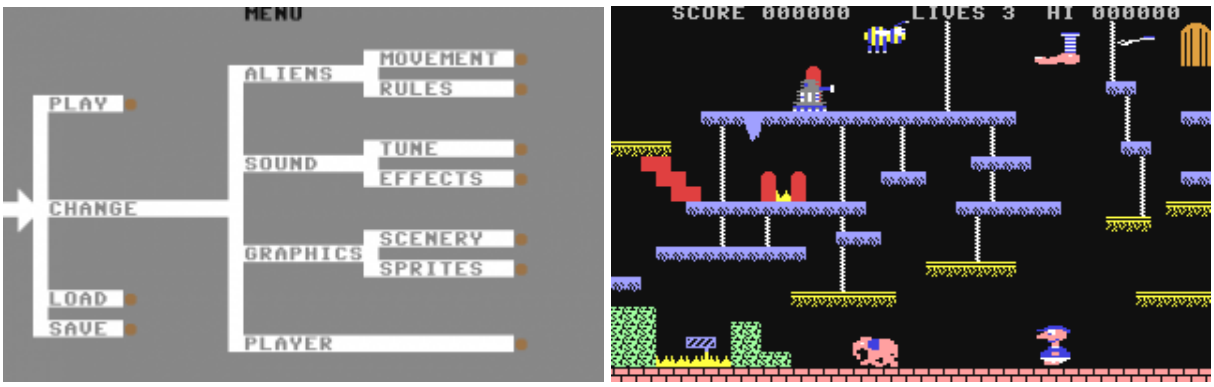
Parmi les *« Construction Set »* qui ont suivi les traces de ***Bill Budge***, *Shoot'Em-Up Construction Kit* ***(Sensible Sofware, 1987)*** fait partie des rares outils suffisamment populaires pour avoir été utilisé par des professionnels. En effet, la plupart des amateurs qui créaient des jeux avec ces outils s'empressaient de les diffuser gratuitement autour d'eux, idéalement par le biais de *BBS*[50] ou de magazines spécialisés. Certains professionnels faisaient de même tout en commercialisant leurs réalisations, à l'image de la société italienne ***System Editoriale s.r.l***. *Shoot'Em-Up Construction Kit* propose un ensemble d'éditeurs permettant de créer les différents aspects d'un jeu d'action et de tir. Un éditeur de décor permet de dessiner différents « blocs », qui pourront ensuite être agencés de manière à créer un gigantesque arrière-plan défilant. Ce même éditeur graphique permet aussi de dessiner les différentes étapes d'animations qui composent les « objets » du jeu *(joueur, ennemis, bonus, projectiles…)*. Enfin, un éditeur de niveaux permet de disposer les différents ennemis le long du décor, en les groupant éventuellement afin de former les « vagues » caractéristiques de ce genre de jeu vidéo. Une fois le jeu terminé, il est possible de le sauvegarder sous forme d'un fichier exécutable autonome qui permet de le distribuer librement.

[50] Un **B**ulletin **B**oard **S**ystem est un ordinateur auquel d'autres ordinateurs peuvent se connecter par le biais d'un modem. Un fois connecté à un BBS par le réseau téléphonique, un utilisateur peut discuter avec d'autres membres, envoyer ou récupérer des fichiers, jouer à des jeux en réseau… Il s'agit en quelque sorte de l'ancêtre d'Internet et de ses nombreux services. Les BBS rencontreront un certain succès auprès des passionnés d'informatique, à une époque où le « réseau des réseaux » n'était pas encore accessible au grand public (soit de la fin des années 1970 au milieu des années 1990).

A noter également que tous ces « *Construction Set* » ont suffisamment marqué les Game Designers amateurs dans les années 1980 pour être aujourd'hui l'objet de « remakes ». Par exemple, *SEUCK* ***(Dream Design Entertainment, 2008)*** et *Shoot'em Up Game Builder* ***(Dan Richardson, 2007)*** sont des versions modernisées de *Shoot'em Construction Kit*.

1.3 THE GAMES CREATOR, A LA FRONTIERE ENTRE AMATEURS ET PROFESSIONNELS

Bien que relativement méconnu, le logiciel *The Games Creator* ***(David Darling & Richard Darling, 1984)*** illustre parfaitement la frontière tenue qu'il peut y avoir entre Game Designers amateurs et professionnels pour l'utilisation d'usines à jeux. A l'origine, ce programme a été inventé sous le nom *Games Designer* par ***David Darling & Richard Darling***. Commercialisé pour le *Commodore Vic-20* par ***Galactic Software***, société fondée par les deux frères, ce logiciel bénéficie alors d'une diffusion plus que confidentielle. Mais la carrière des deux programmeurs anglais de n'arrête pas là. Dans les années 80, le Royaume-Uni connaît une explosion de la création vidéoludique grâce à l'arrivée de la micro-informatique (Donovan, 2010). Ainsi, un bref passage chez l'éditeur ***Mirrorsoft*** permet aux frères ***Darling*** d'améliorer leur logiciel en le portant sur *Commodore 64* ***(Commodore, 1982)***. Cet outil sera alors commercialisé à nouveau sous le nom *The Games Creator* en 1984. Il sera réédité en 1985 sous le même nom par ***Mastertronic***, éditeur pour lequel les frères ***Darling*** travaillèrent jusqu'en 1986. Après leur départ, ils fondent une nouvelle société, ***Code Masters***, plus tard rebaptisée ***Codemasters***. Cette société est aujourd'hui un studio de développement de jeu vidéo reconnu, notamment grâce à sa série *Colin McRae Rally* ***(1998-2011)***. Les frères ***Darling*** furent même décorés « *Commanders of the Order of the British Empire* », l'équivalent britannique de la Légion d'Honneur, en récompense de leur brillante carrière dans l'industrie vidéoludique. Pourtant, à ses débuts, la société ***Codemasters*** se contente de produire des jeux à petit budget sur *Commodore 64*. Pour réaliser ces titres, les ***Darling*** utilisent leur propre usine à jeux, dont les divers éditeurs permettent de créer rapidement et simplement des jeux de tir, d'action et de plateforme. Après avoir ainsi commercialisé de nombreux titres, ***Codemasters*** rééditera cet outil pour la quatrième et dernière fois, sous le nom *Creations*, en 1987.



53. *The Games Creator* : menu principal *(gauche)* et exemple de jeu réalisable *(droite)*

Le fonctionnement de cet outil dans version *Commodore 64* est on ne peut plus simple. Le logiciel s'ouvre sur une arborescence faisant office de menu principal. L'utilisateur y dirige un curseur comme s'il déplaçait un personnage dans un labyrinthe, à la différence que chacune des terminaisons de l'arborescence lance un éditeur permettant de créer un « morceau » de jeu vidéo. L'éditeur de dessin permet de créer des « blocs », qui sont rangés en trois catégories : *décor de fond, obstacle, et piège*. L'utilisateur peut ensuite utiliser ces blocs pour dessiner le niveau du jeu, en les assemblant à la manière d'un puzzle. Selon la catégorie à laquelle appartient le bloc, il aura une influence différente dans le jeu : soit de bloquer le mouvement du joueur, soit de le tuer, ou aucune s'il s'agit d'un décor. Bien que les niveaux ne peuvent être dessinés que sur un seul écran, il est possible d'appliquer un

« scrolling horizontal » au niveau, qui défilera alors en boucle. L'éditeur de dessin permet ensuite de créer des étapes d'animations pour les deux catégories d'éléments de jeu : le *« joueur »* et les *« aliens »*. La première catégorie d'élément est l'avatar contrôlé par le joueur, dont il est possible de configurer quelques paramètres tels que la vitesse de déplacement ou la capacité de tir, dans un menu dédié. Les *« aliens »* sont tous les autres objets actifs à l'écran. Selon les paramètres qui leur auront été appliqués dans le menu de configuration idoine, il s'agira d'ennemis à éviter, d'objets à récolter, ou d'une porte qui permet de passer au niveau suivant. Un dernier éditeur permet enfin de créer et associer des sons aux différents éléments du jeu. Une fois les différentes « parties » du jeu ainsi créées, il est possible de tester le jeu résultant, puis de le sauvegarder sous forme autonome. La première version de cet outil, *Games Designer*, fonctionne sensiblement de la même manière. Précisons tout de même que *Games Designer* subit les limites techniques du *Commodore Vic-20*, dont les capacités graphiques sont bien plus réduites que le *Commodore 64*. Une autre différence notable entre les deux versions de cet outil est le menu principal d'édition. Dans *The Games Creator* il s'agit d'une arborescence que l'on peut parcourir de manière non-linéaire. Il est ainsi possible de travailler sur chacun des aspects du jeu à tout moment, et dans l'ordre souhaité. Dans *Games Designer*, ce menu principal n'existe pas. Une fois lancé, le logiciel propose tout simplement chacun des éditeurs les uns après les autres, dans un ordre immuable : *éditeur de dessin et d'objets, éditeur de niveau, éditeur de son, assemblage du jeu final*. Le processus de création est donc relativement fastidieux pour corriger un aspect précis du jeu, car il faut relancer le programme, puis quitter les éditeurs les uns après les autres jusqu'à tomber sur celui que l'on souhaite utiliser. Il est ici intéressant de noter que la première version de l'outil forçait les différentes étapes du processus de création d'un jeu vidéo, alors que les versions suivantes laissent l'utilisateur libre de sa méthodologie de travail. Cela nous renvoie indirectement à la question de la formalisation d'une « série d'étapes » du processus Game Design observée avec les outils théoriques *(p.65)*.

Au final, cet outil perpétue la recette imaginée par *Pinball Construction Set* ***(Bill Budge, 1983)*** : un ensemble d'éditeurs spécialisés pour chaque aspect du jeu *(graphisme, son, niveau, règles de jeu…)*, et la possibilité d'assembler le tout pour créer un nouveau jeu vidéo autonome. En explorant un genre de jeu différent de son prédécesseur, il contribue à améliorer un principe de fonctionnement qui sera repris par de nombreux autres outils, tels que *Shoot'Em-Up Construction Kit* ***(Sensible Sofware, 1987)*** pour les jeux de tir ou *Game-Maker* ***(Recreational Software Design, 1991)*** pour les jeux de plateforme. De plus, l'histoire de cette usine à jeux aux multiples noms illustre parfaitement la frontière ténue entre les mondes professionnel et amateur de la création vidéoludique. Au fond, ces deux mondes sont à la recherche d'outils leur permettant de simplifier, et donc d'accélérer, le processus de création d'un jeu vidéo. Si aujourd'hui la connaissance technique requise pour utiliser les outils professionnels les mets souvent hors de portée des amateurs *(p.46)*, dans les années 80 les premières usines à jeux disponibles étaient accessibles à tous.

1.4 ZZT, L'IMPORTANCE DE LA CREATION COMMUNAUTAIRE

Si les usines à jeux des années 80 rencontrent un certain succès grâce au développement de la micro-informatique personnelle, la dimension communautaire des concepteurs de jeux amateurs est alors relativement peu développée. En effet, faute d'Internet, seules des communautés « locales » se créent, que ce soit dans la cour d'une école ou autour de serveurs *BBS* locaux. Avec le développement des serveurs *BBS* internationaux, suivi de celui d'Internet, les Game Designers amateurs voient arriver des moyens de communications plus efficaces pour échanger leurs créations. Comme nous venons de le voir, les premières usines à jeux sont principalement le fruit de passionnés ou de petites sociétés d'édition. Dans les années 1990, l'avènement d'Internet permettra à ces deux catégories de développeurs de distribuer plus facilement leurs produits sous une même bannière : le *« shareware »*. Ce mode

de distribution particulier, hérité du monde du logiciel utilitaire, consiste à diffuser une version gratuite mais limitée d'un programme (Callahan, 2000; Knopf & Ford, 2000). Les usagers peuvent librement télécharger cette version limitée par le biais d'Internet, de serveurs *BBS* ou la trouver sur les cédéroms accompagnant des magazines spécialisés. Si le programme leur plait, ils peuvent acheter la version complète en la commandant directement à son concepteur. Facile à mettre en place et peu onéreux, ce mode de distribution a séduit de nombreux particuliers et petites structures qui commercialisent des jeux ou des logiciels utilitaires. Parmi les jeux vidéo les plus connus de la sphère du *« shareware »* se trouvent *Wolfenstein 3D* ***(id Software, 1992)***, *Doom* ***(id Software, 1993)*** ou encore *Duke Nukem 3D* ***(3D Realms, 1996)***. A noter que tous ces titres sont livrés avec un éditeur de niveaux et sont des piliers de la pratique du « modding » *(p.310)*.

Au sein des nombreux logiciels commercialisés par le biais du *« shareware »*, nous retrouvons des usines à jeux, à l'image de *ZZT* ***(Tim Sweeney, 1991)***. Ce logiciel permet de créer facilement des jeux d'action-aventure utilisant des graphismes *ASCII*[51]. Grâce aux *BBS* et à un Internet naissant, les Game Designers amateurs utilisant cette usine à jeux se sont structurés en une véritable communauté de créateurs, vouée à l'échange de réalisations et de conseils. Le nom *ZZT* a d'ailleurs été choisi pour que le logiciel apparaisse en dernière position sur les listings *BBS*, le rendant ainsi plus facile à trouver. La création de jeux avec *ZZT* est on ne peut plus simple. L'outil principal est un éditeur de niveau, qui permet de créer des *« tableaux »* pouvant être liés entre eux par des *« passages »*. Comme tous les graphismes sont tirés de la table *ASCII*, il n'y a pas besoin de dessiner les objets de jeu, mais juste de choisir le caractère et la couleur souhaitée dans une liste. Le jeu propose de nombreux objets préprogrammés, comme des ennemis ou des objets à récolter. Mais il permet également de créer ses propres comportements à partir d'un langage de script propriétaire. Un élément baptisé *« object »* apparaît dans la liste des objets utilisables pour construire des niveaux. En le plaçant sur la scène, l'utilisateur peut choisir son apparence graphique, mais aussi son comportement, en utilisant un langage de script dédié. Ce dernier suit le paradigme *« Orienté Objet »*, et propose une trentaine de méthodes allant de *« déplacer l'objet »* à *« tirer un projectile »* en passant par la modification de variables ou l'envoi de messages entre objets. Ce langage permet ainsi d'appréhender une forme simplifiée de programmation évènementielle liée au contexte du jeu. Au final, *ZZT* semble pensé pour faciliter au maximum la création de jeux d'action-aventure avec ses objets préprogrammés, tout en conservant la possibilité d'enrichir considérablement lesdites aventures par le biais d'un langage de programmation. Par contre, *ZZT* ne permet pas d'exporter directement ses jeux sous forme de fichier exécutable. Pour jouer à un jeu créé avec *ZZT*, il faut le charger dans le logiciel. Mais ce dernier étant librement disponible par le biais du *« shareware »*, nous pouvons considérer qu'il s'agit encore de *« création de nouveaux jeux autonomes »* et non de *« modification de jeux existants »*, même si cet outil se trouve à la frontière des deux critères de notre modèle classificatoire *(p.160)*.

[51] Acronyme de **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. Il s'agit d'une norme informatique pour l'encodage des caractères. Avant l'apparition de moteurs graphiques performant, il était possible de détourner les fonctions d'affichage textuel d'un ordinateur pour simuler des dessins, en utilisant notamment les caractères spéciaux définit dans la norme ASCII.

54. *ZZT* (gauche) et *Official Hamster Republic Role Playing Game Creation Engine* (droite)

En plus d'une suite nommée *Super ZZT* ***(Epic Megagames, 1992)***, *ZZT* est rapidement suivi par d'autres logiciels similaires, tels que *MegaZeux* ***(Gregory Janson, 1994)***. Cet outil est originellement pensé comme une simple amélioration de *ZZT*, son auteur ***Gregory Janson*** étant un Game Designer amateur issu de la communauté dédiée à cette usine à jeux. Souhaitant dépasser les nombreuses limites techniques de *ZZT*, il commença par proposer un outil complémentaire du nom de *Super Tool Kit* ***(Gregory Janson, 1993)***. Il réalisera ensuite sa propre usine à jeux inspirée par *ZZT* : *MegaZeux*. Cet outil connaîtra une carrière plus longue que son modèle, en partie grâce à son passage en open-source en 1999. Le développement de cette usine à jeux a alors été poursuivi par une communauté d'amateurs passionnés. Aujourd'hui, de nombreux jeux continuent à être créés avec *MegaZeux*, grâce une communauté d'utilisateurs restreinte mais toujours active plus de quinze ans après la sortie du logiciel. *Official Hamster Republic Role Playing Game Creation Engine* ***(James Paige, 1997)*** a connu un destin similaire. D'un logiciel payant, il est devenu un logiciel gratuit avant d'atteindre le statut open-source. Il continue donc à être régulièrement mis à jour, ayant déjà permis à ses utilisateurs de créer plusieurs centaines de jeux de rôle et d'aventure en 2D.

1.5 M.U.G.E.N., LE HACKING COMME ALTERNATIVE A L'OPEN-SOURCE

Ce phénomène de communautés de Game Designers amateurs qui se forment autour d'un logiciel donné par le biais d'Internet est observable dans de nombreux cas, même pour des logiciels qui ne sont pas open-source. Ainsi, *M.U.G.E.N.* ***(Elecbyte, 1999)*** est une usine à jeux dédiée à la création de jeux de combats. Distribuant les premières versions gratuitement à des fins de bêta-test, l'éditeur du logiciel fédéra rapidement une large communauté de passionnés. Alors que de nouvelles versions bêta sortaient régulièrement, ***Elecbyte*** cessa son activité en 2001, laissant une large communauté d'utilisateurs abandonnée avec un logiciel fermé, qu'elle ne pouvait donc plus faire évoluer. Qu'a cela ne tienne, certains membres de cette communauté hackèrent *M.U.G.E.N.* pour continuer à le faire évoluer ***(Rouhei, 2004)***, avant que de nombreuses tentatives de clonage ne voient le jour : *OpenMugen* ***(2006)***, *Infinity Cat* ***(SofiyaCat, 2007)***, *Paintown* ***(Jon Rafkind, 2007)***, *DXG* ***(Thkware, 2008)***, *xnaMugen* ***(frantzx, 2009)***, *I.K.E.M.E.N.* ***(Suehiro, 2009)***, *JMugen* ***(lknhiayi, 2009)*** et *Shugendo* ***(Sakirsoft, 2010)***. Tous ces outils amateurs avaient pour objectif de recréer une usine à jeux de combats compatible avec le standard définit par *M.U.G.E.N.* Si ces divers projets furent plus ou moins aboutis, le fait que les créateurs originaux reprirent leur activité après dix ans d'absence en créant *M.U.G.E.N. 1.0* ***(Elecbyte, 2011)*** ressouda cette communauté derrière un seul logiciel.

55. *M.U.G.E.N. 1.0* (gauche) et *Fighter Factory* (droite)

Pour autant, ces initiatives visant à créer un logiciel émulant les capacités d'une usine à jeux populaire n'auront pas été vaines. Elles ont donné naissance à des outils facilitant la création de jeux de combats avec *M.U.G.E.N.*. De base, cette usine à jeux est fournie avec plusieurs éditeurs permettant de créer les différentes parties d'un jeu de combat. Seul problème, la plupart de ces outils sont loin d'être intuitifs, car ils s'appuient sur des fichiers textes que l'utilisateur doit rédiger selon un formalisme précis. Ainsi, un outil permet de compresser les images et sons importés dans des formats lisibles par l'usine à jeux, tandis que la création de niveau, la programmation de personnages ou même la création d'animations s'effectue en éditant des fichiers textes. Pour la programmation des personnages, ils sont considérés comme des *« machines à états finis »* : chaque personnage du jeu est constitué d'une collection d'états *(attente, vient de recevoir un coup, en train de faire une attaque spéciale…)*. Il faut alors définir, pour chacun des états, un numéro d'animation, puis régler divers paramètres tels que les forces de déplacements du personnage ou la puissance des coups qu'il est en train de porter. Il est ensuite possible d'ajouter des *« conditions »* permettant de contrôler les changements d'états du personnage. Certaines de ces *« conditions »* introduisent les manipulations complexes sur l'interface entrante qui caractérisent ce genre vidéoludique. La création d'animation passe par un fichier texte séparé, dans lequel il faut noter des numéros d'images dans leur ordre d'apparition, assortie d'une durée d'affichage pour chaque étape d'animation. Le seul *« éditeur visuel »* fourni permet de dessiner les zones de collisions pour chaque étape d'animation du personnage, une fois que le fichier d'animation a été créé dans un éditeur de texte. Si ces outils quelque peu rudimentaires n'ont pas empêché de nombreux passionnés de se lancer dans la création de jeux de combats, la communauté de Game Designer amateurs utilisant *M.U.G.E.N.* s'est considérablement agrandie avec l'arrivée d'outils « officieux » simplifiant ce processus de création. Par exemple, *Fighter Factory* ***(Ramon do Rosário Saldanha, 2009)*** propose plusieurs « éditeurs visuels » qui facilitent grandement la création d'animations, de niveaux, et des différents états des personnages. Cet outil génère ensuite des fichiers textes qui seront interprétés par le moteur à la base de *M.U.G.E.N.*, permettant ainsi de créer plus facilement des jeux de combats.

Ce genre d'initiative, qui pousse des amateurs à créer des éditeurs externes remplaçant les éditeurs livrés en standard avec une usine à jeux, n'est pas limité à cet exemple illustratif. On le retrouve par exemple dans des outils destinés à la création de jeux éducatifs, tels que avec *Adventure Author* et *Flip* *(p.225)*, qui remplacent les outils de *Neverwinter Nights 2* ***(Bioware, 2006)***, ainsi qu'avec *Virtuoso* *(p.220)* qui transforme ceux fournis avec *Half-Life 2* ***(Valve Software, 2004)***.

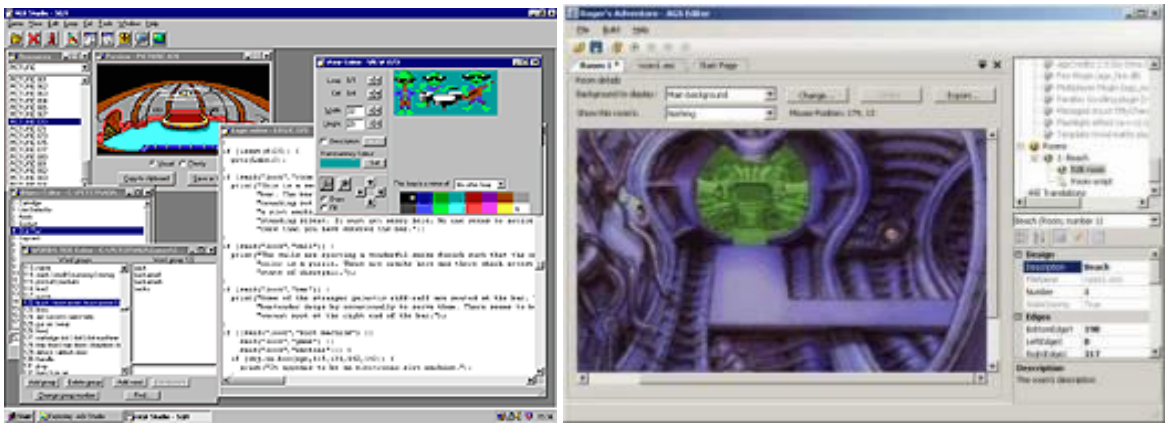
Si le jeu d'aventure textuel est vraisemblablement le genre le plus propice à l'apparition d'usines à jeux spécialisées *(p.166)*, son évolution qu'est le genre du jeu d'aventure graphique n'est pas en reste. Dans les années 80 et 90, les aventures à base de texte furent progressivement éclipsées par d'autres jeux d'aventure, richement illustrés et animés. Les deux sociétés leaders du secteur étaient ***Sierra*** et ***LucasArt***. Afin de pouvoir produire rapidement de nombreux titres, ces sociétés ont développé en interne des outils leur permettant d'accélérer la réalisation de jeux d'aventure graphique (Rob Smith, 2008). Concrètement, il s'agit de *« middleware »* *(p.46)*, même si à l'époque ce terme n'était pas utilisé au bénéfice de l'appellation plus générale de « moteur de jeu ». Ces outils se présentent sous la forme d'un ensemble d'éditeurs permettant de créer des décors ou des personnages animés, assorti d'un langage de script propriétaire permettant de définir l'interactivité des différentes scènes du jeu. Ces « moteurs de jeu » sont similaires aux usines à jeux de l'époque, à la différence qu'elles n'étaient pas diffusées au grand public mais utilisées uniquement en interne par chaque société.

Du côté de chez ***Sierra***, le premier moteur de jeu s'appelle *Adventure Game Interpreter (AGI)* et fut utilisé entre 1983 et 1989. Il fut utilisé pour créer des grands classiques du jeu d'aventure comme *King's Quest: Quest for the Crown* ***(Roberta Williams, 1984)***, *Space Quest: The Sarien Encouter* ***(Mark Crowe & Scott Murphy, 1986)*** et *Leisure Suit Larry in the Land of the Lounge Lizards* ***(Al Lowe, 1987)***. Il a également permis la création d'ancêtres des Serious Games *(p.33)*, à l'image du jeu éducatif *Donald Duck's Playground* ***(Al Lowe, 1984)***, qui traite du calcul et de la logique. Ce moteur n'a jamais été diffusé auprès du grand public. Pourtant, plus de dix ans après sa dernière utilisation par ***Sierra***, des amateurs ont entrepris de hacker les jeux qu'il fait tourner pour proposer des outils permettant de créer de nouveaux titres. Ainsi, *AGI Game Studio* ***(Peter Kelly, 2000)*** est une usine à jeux permettant de créer de nouveaux jeux d'aventure tournant sur le moteur *AGI*. Bénéficiant de l'évolution technologique et ergonomique, ce logiciel est bien plus facile à utiliser que les outils originellement créés par ***Sierra***. Un destin similaire a été connu par le successeur d'*AGI*, nommé *Sierra's Creative Interperter (SCI)*. Utilisé de 1988 à 1996 par ***Sierra*** afin de continuer ses séries phares, ce moteur est plus complexe : il a connu de nombreuses versions incompatibles entre elles. En conséquence, le travail des hackers souhaitant créer des outils officieux relatifs à *SCI* est bien plus ardu. Par exemple, *SCI Studio* ***(Brian Provinciano, 2001)*** et *SCI Companion* ***(Troflip, 2007)*** ne permettent de créer des jeux que pour la première version du moteur *SCI*, communément appelée *SCI0*. Des outils existent pour travailler sur des versions plus récentes du moteur, mais à ce jour ils permettent seulement d'analyser des jeux existants, et non d'en créer de nouveaux. S'il reste donc du travail pour hacker les moteurs de ***Sierra***, en comparaison celui de ***Lucasarts*** est un terrain inexploré. Le moteur *Script Creation Utility for Maniac Mansion (SCUMM)* fut utilisé entre 1987 et 1998 pour créer d'autres grands classiques du jeu d'aventure, tels que *Maniac Mansion* ***(Ron Gilbert & Gary Winnick, 1987)***, *Indiana Jones and the Last Crusade* ***(Ron Gilbert & al., 1989)***, *The Secret of Monkey Island* ***(Ron Gilbert & al., 1990)*** ou *Sam & Max Hit the Road* ***(Sean Clark & al., 1993)***. Le succès rencontré par ces jeux a rapidement entraîné des projets d'usines à jeux capables de créer de nouveaux titres pour ce moteur, tels que *SCRAMM* ***(Jimmi Thogersen, 1999)*** et *Glumol* ***(Bob & Alex, 2000)***. Malheureusement ces deux outils ne virent jamais le jour. Ils restèrent à l'état d'annonce, bien que leurs développeurs respectifs communiquaient beaucoup sur leur avancement. Ces deux logiciels sont restés célèbres dans l'histoire des usines à jeux comme des *« vaporwares »*[52]. Ils sont mêmes devenus des références culturelles ironiques : un projet d'usine à jeux aux fonctionnalités ouvertement ambitieuses sera souvent comparé à *SCRAMM* ou à *Glumol* au détour d'un forum. Il faudra

[52] Ce terme désigne un logiciel resté à l'état d'annonce (« logiciel vaporeux »), car n'ayant jamais été diffusé malgré le fait que ses développeurs aient abondamment et longuement communiqué sur ses fonctionnalités.

attendre quelques années supplémentaires avec la sortie de *ScummC* ***(Alban Bedel, 2006)*** pour voir l'ébauche d'un logiciel permettant de créer des jeux compatibles avec le moteur *SCUMM*. Ce projet sera malheureusement abandonné après la sortie de versions bêta prometteuses. Les espoirs de cette communauté de créateurs de jeux d'aventure reposent aujourd'hui sur *ScummGen* ***(sronsse, 2009***), toujours en cours de développement.

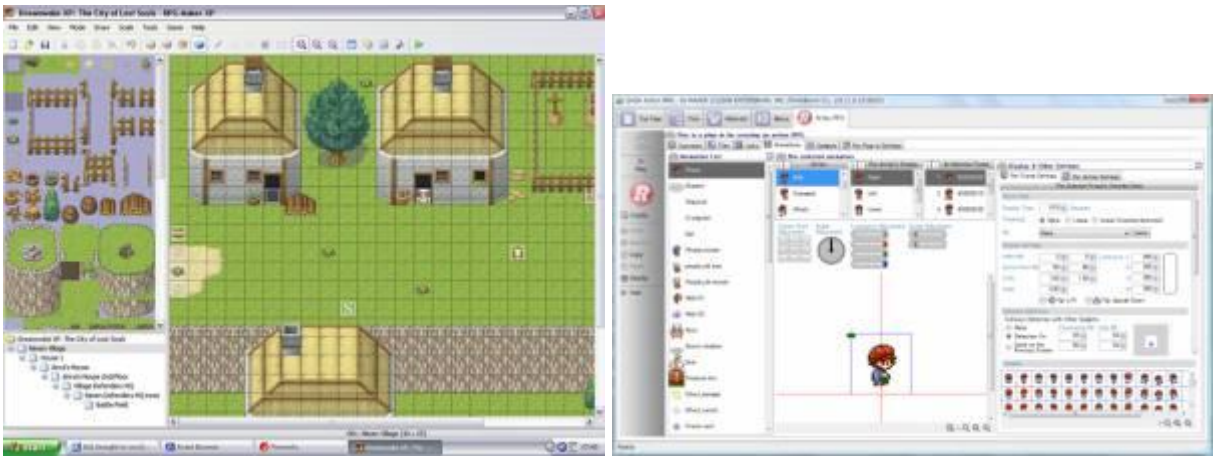
Heureusement, les usines à jeux spécialisées dans le genre des aventures graphiques ne s'arrêtent pas à ces nombreuses tentatives de « *hacking* » des moteurs de jeux commerciaux. En effet, un total de 50 outils techniques représente cette catégorie dans notre corpus. Par exemple, le logiciel *World Builder* ***(William C. Appleton, 1986)*** rencontra un énorme succès sur *Macintosh*, et permit la création de nombreux jeux d'aventures uniquement destinés à cette machine. Côté *PC*, les logiciels les plus célèbres sont *Adventure Game Studio 3.0* ***(Chris Jones, 2008)***, *Wintermute Engine* ***(Jan Nedoma, 2003)***, *Point & Click Development Kit 2* ***(Ben Maas, 2006)*** et *Visionaire 3.0* ***(Thomas Dibke, 2008)***. Bien que récents, ces différents outils ont en général une histoire assez longue, et changent parfois de nom au cours de leur évolution. Ainsi, *Adventure Creator* ***(Chris Jones, 1997)*** est la première version d'*Adventure Game Studio*, alors que *Visionaire 2d* ***(Thomas Dibke, 2002)*** marque les débuts de *Visionaire*. La plupart des usines à jeux d'aventure graphique sont destinés à la création de jeux en 2D, à l'exception de certains logiciels comme *3D Adventure Studio* ***(Marten van der Honing, 2003)*** qui propose une représentation 3D. Globalement, tous ces logiciels fonctionnent plus ou moins de la même manière. Ils permettent tout d'abord la création « d'objets », auxquels peuvent être assignés différentes animations. Une fois les divers éléments de jeux ainsi créés, l'utilisateur peut les agencer pour créer les nombreuses « scènes » qui composeront son jeu d'aventure. Vient ensuite la phase de la création de l'interactivité, qui s'appuie généralement sur un langage de script propriétaire. Pour chaque élément dans une scène, l'utilisateur pourra programmer une série de « réactions » aux actions du joueur. Prenons par exemple un objet « livre ». Le créateur du jeu pourra décider que, si le joueur effectue l'action « *prendre le livre* », l'objet « *livre* » s'ajoutera à la variable « *inventaire* » de l'objet « *avatar du joueur* ». Mais si le joueur effectue simplement l'action « *lire le livre* », le concepteur du jeu programmera alors une autre réaction qui affiche une image en plein écran représentant une page ouverte de ce « *livre* ».



56. *AGI Game Studio* (gauche) et *Adventure Game Studio* (droite)

Les usines à jeux d'aventure graphique ne sont pas réservées au seul secteur du divertissement, puisque certains logiciels de création de Serious Games sont aussi spécialisés dans ce genre. Ils fonctionnent globalement de la même manière que les outils que nous venons de présenter, à l'exception du fait qu'ils sont ouvertement décrit comme permettant de créer des Serious Games. Du côté de la 2D, nous recensons *<e-Adventure>*, créé par un laboratoire de recherche informatique de l'université de ***Madrid*** *(p.222)*. Pour les jeux en 3D, *Thinking Worlds* est quant à lui le fruit d'une société privée spécialisée dans la formation et le e-learning *(p.214)*.

Un dernier exemple du vaste monde des usines à jeux spécialisées permet d'illustrer un autre type d'influence que peuvent exercer les communautés d'amateurs qui se forment spontanément autour de certains logiciels. A la base, la série des *RPG Maker* ***(ASCII, 1992-2010)*** est une famille de logiciels permettant de créer des jeux de rôles d'un genre similaire à ceux de la série *Final Fantasy* ***(Squaresoft, 1987-2010)***. Ils s'agit de logiciels commerciaux disponibles sur une large variété de support : *MSX*, *PC-9801*, *PC (Windows)*, *Super Nintendo*, *Gameboy*, *Playstation*, *Saturn*, *Playstation 2*, *Nintendo DS* et même téléphone mobile. Uniquement disponibles en langue japonaise, ces usines à jeux ne sont pas commercialisées en dehors de leur pays d'origine. Pourtant, par le biais d'Internet, ces logiciels connaissent une carrière internationale fulgurante. En effet, des amateurs les ont traduits en anglais, avant de les distribuer gratuitement sur Internet. En dépit du statut illégal de cette distribution, de très nombreux Game Designers amateurs ont alors utilisé ces logiciels, s'organisant en communautés de créateurs. Les versions pour *Windows* de cet outil, *RPG Maker 95* ***(ASCII, 1995)***, *RPG Maker 2000* ***(Enterbrain, 2000)***, *RPG Maker 2003* ***(Enterbrain, 2002)***, *RPG Maker XP* ***(Enterbrain, 2004)*** et *RPG Maker VX* ***(Enterbrain, 2007)*** font d'ailleurs partie des usines à jeux spécialisées les plus populaires chez les Game Designers amateurs. En dépit de la très large circulation de versions pirates traduites par des amateurs, cette exposition internationale inattendue a permis à ***ASCII/Enterbrain***, la société créatrice des *RPG Maker*, de traduire et de commercialiser officiellement d'autres usines à jeux de leur catalogue. Par exemple, *Action Game Maker* ***(Enterbrain, 2009)***, qui permet de créer des jeux de plateforme, de tir, et d'action-aventure, fut officiellement traduit en anglais et commercialisé sous le nom de *Indie Game Maker*.



57. *RPG Maker XP* (gauche) et *Action Game Maker* (droite)

Toutes les versions de *RPG Maker* pour *PC (Windows)* sont globalement basées sur le même principe de fonctionnement. Tout d'abord, l'utilisateur commence par créer des *« cartes »* en utilisant « l'éditeur visuel » dédié. Cet éditeur de niveau est couplé à une librairie de graphismes livrés avec le logiciel, permettant de créer des jeux sans forcément avoir besoin d'en dessiner tous les éléments. Il reste cependant possible d'importer des images externes. Comme toute usine à jeux spécialisée, cet outil préprogramme certains aspects du jeu. Ici, ce sont les règles liées au contrôle d'une équipe de héros, qui caractérisent le genre des jeux de rôles à la japonaise, qui sont automatiquement construites par le logiciel. Le créateur amateur devra par contre créer l'interactivité de son jeu en disposant des *« évènements »* au sein des niveaux de jeu. Un *« évènement »* est une sorte « d'objet » interactif. Il se voit doté d'une représentation graphique, qui peut être dessinée à la main ou issue des libraires graphiques de base : personnage, élément de décor… Un *« évènement »* possède des *« conditions de déclenchement »* et des *« actions »*, qui sont sélectionnables dans une liste. En général, les *« conditions de déclenchement »* seront évaluées dès que l'équipe de héros contrôlée par le

joueur touchera l'objet *« évènement »* sur la carte. Par exemple, il est possible de créer un *« évènement »* ayant l'apparence d'un personnage qui avance automatiquement vers le joueur. Cet *« évènement »* peut être associé à une *« condition de déclenchement »* qui vérifie le nombre de *« pièces d'or »* possédé par les personnages du joueur. Si les *« pièces d'or »* sont supérieures à un certain seuil, alors le concepteur du jeu peut décider de lancer une action *« commencer un combat »* entre les héros du joueur et le personnage de *« l'évènement »*. Cela permettrait par exemple de créer un ennemi de type « voleur » qui ne s'attaque au joueur que s'il est suffisamment riche. Les autres *« actions »* proposées reflètent les codes de ce genre de jeu, allant de la gestion d'inventaire pour les personnages à la création de dialogues interactifs. Il en va de même pour les *« conditions »*, qui gèrent des variables numériques ainsi que des listes de variables booléennes associées à des noms. Cette fonctionnalité, baptisée *« drapeau »*, permet d'enregistrer rapidement les actions du joueur *(a récupéré une clé bleue, a répondu oui au maire du village, a tué le monstre vert…)*, simplifiant ainsi la création d'aventure riches en rebondissements. Pour les utilisateurs qui se sentiraient limités par ce système « d'éditeur visuel » pour les règles du jeu, il est également possible d'utiliser un langage de script propriétaire dérivé du *Ruby*. Une fois le jeu terminé, il sera exporté sous forme autonome afin de le distribuer. La dernière version en date, *RPG Maker VX*, permet même de commercialiser ses réalisations. Les nombreuses versions pour console de cette famille d'outils sont basées sur un principe de fonctionnement similaire, à deux exceptions près : il n'est pas possible d'importer des images et sons externes, et les jeux créés nécessitent obligatoirement l'achat du logiciel pour être joués. Ces deux limitations sont visiblement d'ordre technique, et liées à l'impossibilité d'importer et d'exporter du contenu sur les consoles pour lesquelles une version de *RPG Maker* est disponible.

Au final, les seize opus de la famille *RPG Maker* permettent de créer relativement facilement des jeux d'aventure riches et profonds. D'après le nombre de résultat retournés par *Google* en saisissant le titre de cette famille d'outil[53], il s'agit apparemment d'une des usines à jeux les plus populaires pour le genre *« aventure & rôle »* couplé à une *« représentation 2D »*. Pourtant, 95 logiciels de notre corpus répondent à ces deux critères. La popularité de *RPG Maker* auprès des créateurs amateurs n'a visiblement pas été arrêtée par le statut illégal des premières versions qui furent distribuées par Internet…

2 LES USINES A JEUX GENERALISTES

Si l'histoire des usines à jeux spécialisées est très riche et intimement liée à la création de communautés de Game Designers amateurs, celle des usines à jeux généralistes lui ressemble fortement en apparence. Elle possède pourtant une différence de fond assez notable. Là où de nombreux amateurs créent des programmes gratuits afin de faciliter la création de jeux d'un genre donné, la grande majorité des logiciels permettant de créer tout type de jeux vidéo sont le fruit de sociétés, et donc distribués commercialement.

2.1 GAMEMAKER, LE PIONNIER

Le premier logiciel que nous recensons dans cette catégorie est édité en 1985 par ***Activision***, sous le nom de *Gamemaker* ***(Garry Kitchen, 1985)***. Ce programme pour *Commodore 64* permet de créer tout type de jeux grâce à une batterie d'éditeurs spécialisé : éditeur d'image pour dessiner des personnages et décor, éditeur de niveau pour les mettre en scène, éditeur de musique. Ces éditeurs ressemblent beaucoup à ceux de la vague des *« Construction Set »* *(p.168)*. En revanche, un nouvel outil fait son apparition en la forme d'un éditeur de logique permettant de créer les règles du jeu. Ces dernières s'appuient sur un langage de

[53] Soit 6 250 000 résultats à la date du 20-08-11

programmation très simple, qui permet uniquement de créer des suites d'instructions de manière linéaire. En effet, l'interface du logiciel est pensée pour être utilisée au joystick, et permet donc de choisir simplement les instructions de programmation dans une liste déroulante (Converse, 1985).

58. *Gamemaker* : éditeur d'image *(gauche)* et éditeur de logique *(droite)*

Pour autant, cette relative simplicité d'utilisation n'empêche pas la création d'une grande variété de jeux vidéo relativement élaborés. Certes, le logiciel arrive avec des limites techniques fortes *(pas de scrolling, nombre de sprites simultanés à l'écran limité, nombre total d'instruction de programmation limité…)*, mais contrairement aux usines à jeux spécialisées, ce logiciel permet de créer tous les types de jeux en vogue à l'époque, voire d'en inventer de nouveaux. Concrètement, là où un logiciel spécialisé impose des parties du jeu « préconstruites » *(notion d'ennemis, de tir, détection de collisions ou modes de déplacement prédéfinis…)*, ici l'utilisateur est face à une « page blanche ». C'est à lui d'inventer l'intégralité des règles de son jeu vidéo, car aucune d'entre elles n'est définie à l'avance. La facilitation de la création passe donc plutôt par la simplification de l'interface du logiciel que par la proposition de « morceaux préconstruits ».

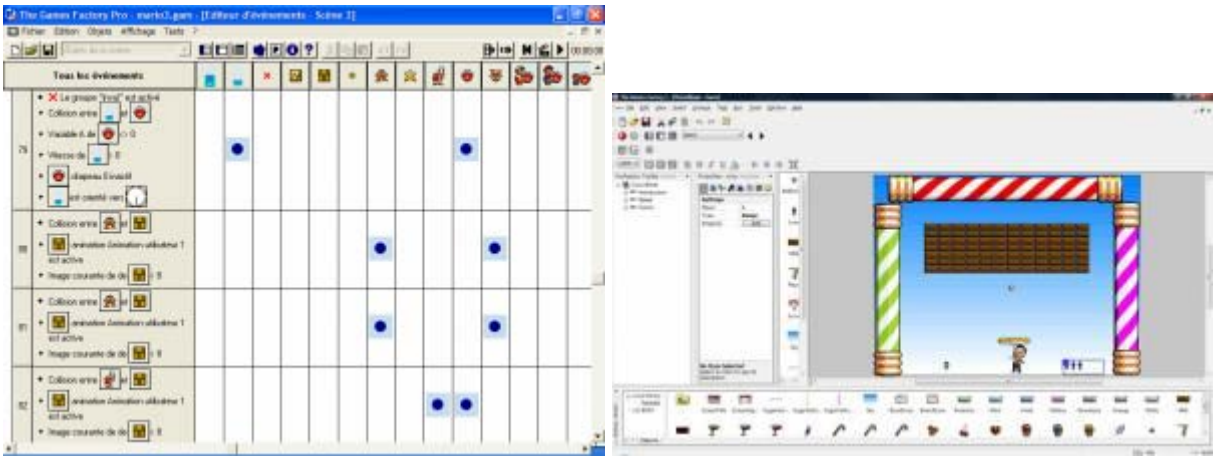
2.2 LA FAMILLE « KLIK », PROGRAMMER SANS ECRIRE

Cette approche est par la suite étendue et améliorée par d'autres développeurs. Parmi ces derniers, la ***Clickteam*** est sans conteste la société qui a apporté la pierre la plus importante à l'édifice des usines à jeux généralistes.

François Lionet a débuté sa carrière par la création de deux usines à jeux génériques, *STOS* ***(Jawx, 1988)*** et *AMOS* ***(Jawx, 1990)***, qui s'appuient sur le langage de programmation *BASIC*. La simplification vient ici de l'approche tout-en-un du logiciel : il est fourni avec des éditeurs graphiques et sonores intégrés. Mais pour définir les règles du jeu, il faut absolument écrire des lignes de codes. Même si le langage *BASIC* reste simple et que la version incluse dans ce logiciel contient beaucoup d'instructions spécifiques au jeu vidéo, le fait de devoir programmer reste une barrière pour nombre d'utilisateurs potentiels de tels logiciels. En s'associant avec ***Yves Lamoureux***, le programmeur français fondera alors la société ***Clickteam*** pour proposer un logiciel s'affranchissant de cette étape. Comme il le raconte lui-même (Pirou, 2010), ***François Lionet*** est parti du constat qu'il existait dans l'univers de la bureautique un logiciel permettant de faire des choses très complexes mais que de nombreuses personnes s'accaparent relativement facilement : *les tableurs*. Les deux inventeurs s'en sont donc inspirés pour remplacer la saisie d'un langage de programmation par un gigantesque tableau permettant de créer des règles de jeux. Cette idée maîtresse donne naissance au logiciel *Klik & Play* ***(Clickteam, 1994)***.

Les jeux créés avec ce logiciel se composent de différents « objets actifs », qui proviennent soit d'une librairie de graphismes livrée avec le logiciel, soit sont dessinés à l'aide d'un outil

intégré. Ces objets peuvent ensuite être agencés dans l'espace pour créer un « niveau ». Une fois le niveau terminé, vient l'étape de la création des règles. Pour cela, un grand tableau est affiché à l'écran avec une colonne pour chaque objet. L'utilisateur commence par choisir une condition dans une liste *(ex : « quand le joueur appuie sur la flèche gauche du clavier »)*, ce qui crée une nouvelle ligne dans le tableau. Il peut ensuite associer une action pour chacun des objets, en réponse à cette condition. Il lui suffit de cliquer dans la case correspondant à un objet, et de choisir une des actions proposées par le logiciel *(ex : « déplacer l'objet vers la gauche »)*. Ainsi, les concepts mobilisés par cette approche sont les mêmes que ceux d'un langage de programmation *(algorithmique, variables…)*, la complexité d'utilisation en moins.



59. *The Games Factory* : éditeur de règles *(gauche)* et *The Games Factory 2* : éditeur de niveaux *(droite)*

Cette façon de « dissimuler » la programmation dans un tableau caractérise ce que les Game Designers amateurs appellent la *famille Klik*. En effet, la ***Clickteam*** a amélioré son concept en continuant à sortir de nouvelles usines à jeux basées sur ce principe : *Click & Create* ***(1996)***, *The Games Factory* ***(1998)***, *Multimedia Fusion* ***(2000)*** ainsi que *The Games Factory 2* ***(2006)*** et *Multimedia Fusion 2* ***(2006)***. Cette famille de logiciel est un des piliers actuels de la création de jeux vidéo par des amateurs, ces derniers étant structurés en communautés. Par exemple, le site *The Daily Click* ***(2002-2011)*** héberge 4132 jeux[54] créés avec cette gamme d'usine à jeux. Si les logiciels de la *famille Klik* sont payants et propriétaires, leur concept innovant a inspiré quelques clones gratuits, tels que *Game Develop* ***(Compil'Games, 2008),*** et open-source comme *Construct* ***(Scirra, 2008)***.

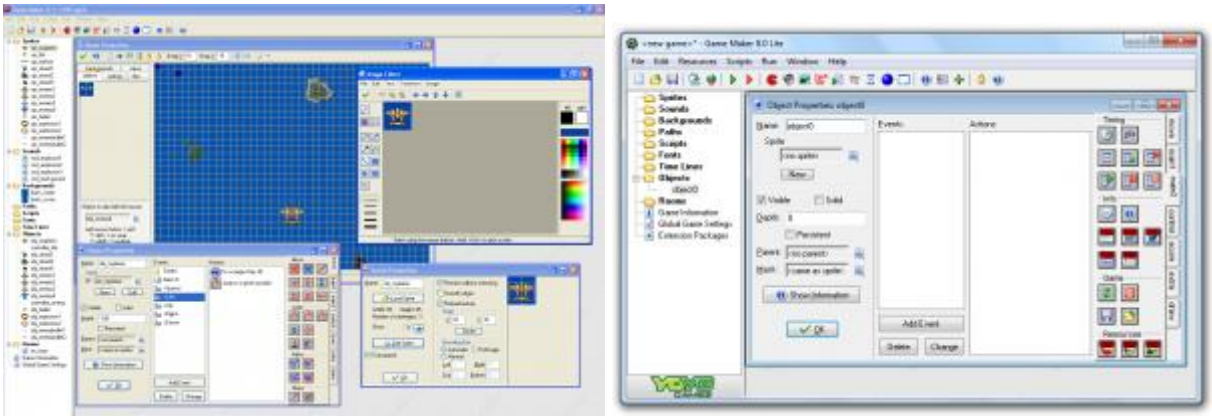
2.3 GAME MAKER, UNE APPROCHE ORIENTEE OBJET

L'idée de permettre la création de règles de jeux, et donc de créer de l'interactivité, sans avoir recours à un langage de programmation n'est pas réservée à une seule famille de logiciel. D'autres ont également tenté cette approche, avec des modalités différentes. Une des initiatives les plus célèbres en la matière nous vient d'un professeur d'informatique, ***Mark Overmars***. Son souhait initial était de s'appuyer sur la conception de jeux vidéo pour enseigner les concepts à la base de la programmation informatique. A ces fins, il a imaginé un logiciel du nom de *Game Maker* ***(Mark Overmars, 1999)***.

D'une manière analogue à la *famille Klik*, cet outil permet de construire des niveaux en agençant des « objets » qui auront été dessinés par l'utilisateur. Si ce logiciel permet d'utiliser un langage de script propriétaire pour définir les règles du jeu, le *GML*, là n'est pas sa philosophie principale, du moins pour les débutants. Cette usine à jeux propose de créer des règles de jeux en associant des « conditions » *(« SI le joueur appuie sur un bouton »)* à des actions *(« ALORS détruire tel objet »)*. Là où *Game Maker* se différencie d'un *The Game*

[54] Relevé le 25-02-11 à partir de http://www.create-games.com/

Factory, c'est dans la manière de « dissimuler » la programmation. *The Game Factory* propose un grand tableau listant toutes les règles, comme un livret de règles de jeu de société. A l'inverse, *Game Maker* se rapproche plutôt de la philosophie de la programmation « Orienté Objet » : il associe directement des règles à chacun des « objets » composant les niveaux. Ainsi, l'utilisateur créé les règles pertinentes pour un objet donné au sein même de cet objet. Nous retrouvons ici un concept proche des *« objets intelligents »* (Kallmann & Thalmann, 1999) : chaque objet possède à la fois ses propriétés *(taille, images…)* et ses méthodes *(règles de jeu)*. Dans la *famille Klik*, les objets ne possèdent que les propriétés, tandis que toutes les méthodes sont référencées dans une liste séparée. Les objets se trouvent donc cantonnés à un simple rôle de conteneurs de données. L'approche *« objets conteneurs de données »*, illustrée par la *famille Klik*, et l'approche des *« objets intelligents »*, présente dans *Game Maker*, sont les deux principales philosophies de création que l'on retrouve dans notre corpus d'usines à jeux généralistes *(p.258)*.



60. *Game Maker 6.0* : éditeur de niveaux *(gauche)* et *Game Maker 8.0* : éditeur de règles *(droite)*

Pour en revenir à *Game Maker*, sa philosophie s'inspirant de l'approche « Orienté Objet » lui permet d'inclure d'autres concepts issus de ce paradigme. Par exemple, l'héritage entre classes d'objets. Dans *Game Maker*, un objet peut hériter d'un autre, et ainsi récupérer par défaut ses propriétés et ses règles. Il est bien évidement possible de créer de nouvelles règles sur l'objet qui hérite. Au final, *Game Maker* remplit l'objectif visé par ***Overmars***, à savoir d'initier les utilisateurs de cet outil à des concepts de programmation informatique. Cet enseignant-chercheur l'utilise d'ailleurs activement lors de ses propres cours d'informatique dans le cadre universitaire (Overmars, 2004). Mais l'université d'***Utrecht*** n'est pas la seule à utiliser cette usine à jeux. A l'image des outils présentés précédemment, cet outil est commercialisé au grand public par Internet. Il y a fédéré une très large communauté de concepteur amateurs, qui s'échangent leurs créations et des conseils. Par exemple, le site *Game Maker Games* ***(2004-2011)*** rassemble à ce jour 3634 jeux vidéo[55]. Cet effort a visiblement inspiré ***YoYoGames***, actuel éditeur de *Game Maker 8.0* ***(Mark Overmars, 2009)***, car son site officiel présente maintenant une section « partage » qui recense déjà 382 titres[56].

2.4 STAGECAST CREATOR, S'ADAPTER AUX NOVICES

Une autre approche de simplification de l'étape de la programmation nous vient d'un centre de recherche privé. Opérationnel de 1986 à 1997, le *Advanced Technology Group (ATG)* fut un laboratoire de recherche interne à la société ***Apple***. Nous lui devons de nombreuses inventions telles que *QuickTime* ***(1991)*** ou *HyperCard* ***(1987)***. A l'initiative de trois ingénieurs de ce laboratoire, ***Allen Cypher***, ***David Canfield*** et ***Kurt Schmucker***, est né un outil du nom de *KidSim*. Ce programme vise à permettre la création de jeux vidéo et de

[55] Relevé le 25-02-11 à partir de http://www.gamemakergames.com/
[56] Relevé le 25-02-11 à partir de http://www.yoyogames.com/

simulations interactives par des enfants. Il sera renommé *Cocoa* pour sa première présentation publique en 1996. Malheureusement, le retour de ***Steve Jobs*** à la tête d'Apple entraîne la fermeture de l'*ATG*, et donc la fin de tous ses projets. Les inventeurs de *Cocoa* quittent alors la société pour continuer le développement de leur logiciel au sein d'une nouvelle structure qu'ils ont fondé, ***Stagecast Software***. *Stagecast Creator*, une version modernisée et améliorée de *Cocoa*, est publiée en 2000. Là où *Cocoa* était un programme spécifique au *Macintosh*, *Stagecast Creator* est réécrit en *Java* afin d'être multiplateforme. Le logiciel rencontrant un certain succès, il sera suivi par *Stagecast Creator 2* ***(2003)***, qui continu d'être mis à jour.

Destiné à une cible enfantine, ce logiciel simplifie au maximum la création de jeux vidéo. En utilisant les objets graphiques livrés avec le logiciel ou en dessinant les siens, l'utilisateur va pouvoir construire une scène de jeu. Il lancera ensuite le programme afin de créer ses différentes règles au fur et à mesure. La différence entre ce logiciel et ceux présentés jusqu'ici est qu'il ne propose pas une liste de « conditions » et « d'actions » pour la création de règles. A la place, l'utilisateur se voit offrir la possibilité de créer les règles de jeux en positionnant graphiquement des éléments. Par exemple, si l'utilisateur veut créer une règle permettant à un personnage de sauter au dessus d'un caillou, il devra tout d'abord placer le personnage à côté dudit caillou. Ensuite, en cliquant sur le bouton de « création de règles », une fenêtre apparaît avec deux visuels identiques étiquetés *« avant »* et *« après »*. Sur ces deux visuels sont affichés le personnage et le caillou. Si sur le visuel *« avant »* l'utilisateur positionne le personnage à gauche du caillou, et que sur le visuel *« après »* il le place au dessus, alors le logiciel créera automatiquement une règle permettant au personnage de passer sur le caillou. Il est bien évidement possible de rajouter des conditions supplémentaires, par exemple la nécessité d'appuyer sur un bouton pour que le personnage saute sur le caillou. Au-delà de cette approche d'utilisation très simple d'accès, la philosophie du logiciel reste proche de celle de *Game Maker*. Chaque objet de la scène de jeu possède ses propres règles, qu'il applique de manière séquentielle pendant que le jeu tourne. Cette usine à jeux étant née dans un laboratoire de recherche, ses concepteurs se sont ouvertement inspirés du concept « d'agents ». Après avoir longuement réfléchi aux différentes approches permettant de programmer sans utiliser un langage dédié, ils concluent que la meilleure approche est de demander à l'utilisateur de « montrer » au logiciel ce qu'il souhaite que les agents « fassent » (David Canfield Smith, Cypher, & Spohrer, 1994). Autre fonctionnalité innovante par rapport à *Game Maker* et aux logiciels de la *famille Klik*, *Stagecast Creator* permet de créer ou modifier les règles d'un jeu pendant qu'il tourne, à condition cependant de le mettre en pause. L'utilisateur peut donc plus facilement affiner les réglages de son jeu sans avoir à le redémarrer à chaque fois, fonctionnalité très utile pour l'évaluation et la correction de prototypes (Fullerton, 2008).

61. *Stagecast Creator 2* : éditeur de niveaux *(gauche)* et éditeur de règles *(droite)*

Bien que plus limité que ses concurrents en terme de puissance, la plus grande simplicité d'utilisation de *Stagecast Creator* le rend accessible à un public plus jeune. Ainsi, le

chercheur ***Jacob Habgood*** a utilisé ce logiciel dans le cadre d'un club d'informatique pour permettre à des enfants de réaliser des Serious Games (Habgood, Ainsworth, & Benford, 2005). En utilisant des activités pédagogiques adaptées[57], ***Habgood*** constate que *Stagecast Creator* est parfaitement accessible à des enfants dès l'âge de 7 ans, alors qu'il recommande plutôt *Game Maker* et *The Games Factory* à partir de 10-11 ans. ***Habgood*** note également que *Stagecast Creator* sera souvent considéré comme « trop limité » par des enfants âgés de plus de 11 ans. Ce retour d'expérience montre à nouveau l'intérêt de l'existence d'une large diversité d'outils techniques dédiés à la création de jeux vidéo *(p.163)*. Pour un pédagogue, cela permet de choisir le logiciel le plus adapté à son public, que ce soit en terme d'âge, de compétence technique ou de temps disponible pour les activités. Mais ce travail montre également l'importance de la présence d'un médiateur dans l'utilisation d'un outil technique de création vidéoludique, et plus particulièrement ici pour la création de Serious Games. En effet, il ne suffit pas de présenter un outil technique à des enfants pour qu'ils créent des jeux vidéo par eux-mêmes. Il faut pouvoir les accompagner, aussi bien sur la démarche de réflexion que sur la manipulation de l'outil, pour leur permettre d'exprimer au mieux leur créativité *(p.253)*.

2.5 VIRTOOLS, LA BRANCHE PROFESSIONNELLE

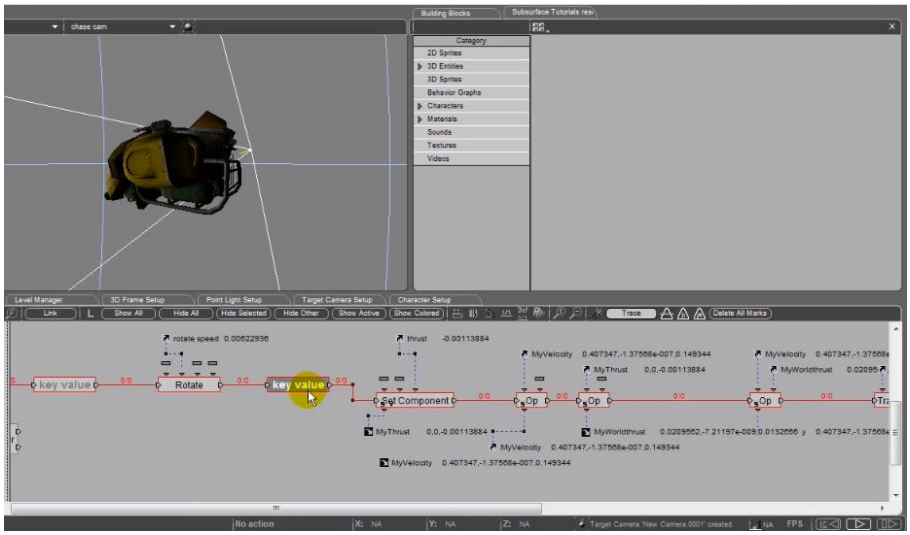
Bien que commerciaux, ces logiciels sont vendus à des prix relativement abordables pour des amateurs : aux alentours de 50€. Il existe d'autres usines à jeux nettement plus onéreuses, ce qui les destine de prime abord aux professionnels. Pourtant, dans leur philosophie visant à simplifier la création de jeux vidéo, tous ces outils partagent une finalité similaire, et constituent en tout cas une source d'inspiration valable quel que soit leur prix de vente. Ainsi, malgré un prix de licence pouvant dépasser les 10.000€, *3DVIA Virtools 5.0* ***(Dassault Systèmes, 2009)*** est une usine à jeux particulièrement intéressante dans son approche de simplification de la programmation. A l'origine, ce logiciel est né sous le nom de *NeMo* ***(NeMo, 1999)*** et visait à rendre accessible la création de jeux vidéo en 3D à toute personne ne sachant pas programmer. Si la création de visuels 3D doit toujours être effectuée à l'aide de logiciels dédiés *(qui ne sont forcément simples d'accès)*, la famille *Virtools* permet bien de créer des règles de jeux sans avoir recours à un langage de programmation.

L'approche choisie est celle de la programmation graphique par l'assemblage de « blocs ». D'une manière analogue à *Game Maker*, l'utilisateur crée des règles pour chacun des objets composant son jeu. Mais au lieu de créer des règles en assemblant des « actions » et des « conditions », il va ici associer des « blocs » de comportements prédéfinis. Par exemple, l'utilisateur peut commencer par poser un bloc *« appui sur une touche de clavier »*, et le configurer pour qu'il soit associé à une touche précise. Ce bloc proposera ensuite deux petits connecteurs de sortie : le connecteur « oui » *(si la touche du clavier est effectivement enfoncé)* et le connecteur « non » *(si elle ne l'est pas)*. A chacun de ces connecteurs, l'utilisateur peut associer d'autres blocs tels que *« faire tourner l'objet »*, *« déplacer la caméra »*... En associant et en configurant ainsi une série de blocs, l'utilisateur crée un *« comportement »* pour l'objet. Philosophiquement, cela reste proche du paradigme de programmation impérative *(si…alors)*, mais la manière de créer la logique informatique est en revanche différente des autres logiciels mentionnés précédemment.

Virtools est livré en standard avec une bibliothèque de « blocs » de comportements, mais il est possible de créer ses propres « blocs » en les programmant en *C++*. Les blocs proposés en standard sont relativement simples, et ne permettent de faire qu'une opération à la fois. Lors de la création d'un jeu, il est souvent nécessaire de créer des suites d'actions que l'on souhaite

[57] Les plans et activités pédagogiques proposées par ***Habgood*** pour *Stagecast Creator* et *Game Maker* sont disponibles ici : http://www.gamelearning.pwp.blueyonder.co.uk/index.html?clubs/resources/resources.htm

réutiliser dans un contexte légèrement différent. Par exemple, une règle telle que *« si j'appuie sur la touche X et que ma réserve de munitions est au moins égale à Y alors je tire un projectile Z devant moi »*, pourra être utilisée plusieurs fois dans un jeu de tir, sous réserve que le concepteur puisse modifier facilement les valeurs *X*, *Y*, et *Z*. La série des *Virtools* propose une approche très intéressante sur ce point. Elle permet à l'utilisateur d'associer plusieurs « blocs de base » et de les encapsuler dans un conteneur plus large *(appelé « bloc de comportement »)* qui possède ses propres paramètres. Il est ainsi possible de créer soi-même des règles élaborées réutilisables sans avoir recours à la programmation en *C++*, du moins tant que les opérations proposées par la liste de *« blocs de base »* suffisent au concepteur du jeu (Krupa, 2003).



62. *3DVIA Virtools 5.0* : éditeur de niveau *(haut)* et de règles *(bas)*

D'une première version en 1999 destinée aux petits studios de création de jeu vidéo, *Virtools* a depuis évolué, autant en prix qu'en fonctionnalités, vers un logiciel qui se destine plutôt aux gros studios et aux sociétés spécialisées dans la réalité virtuelle ou la simulation industrielle. Son approche reste cependant très pertinente pour la création de jeux vidéo par des amateurs, ou par des professionnels ne possédant pas de compétences en programmation informatique. Ainsi, la société ***Onlineformapro*** a conçu une usine à jeux destinée à simplifier la création de Serious Games, *SGTools* *(p.214)*. Comme son nom l'indique, cet outil reprend les concepts à la base de *Virtools* en ce qui concerne la création des règles de jeu sans langage de programmation. Utilisant lui aussi un mode de représentation 3D, il est par contre basé sur la technologie *Flash* ***(1996-2010)***, ce qui le différencie de son modèle. Là où *Virtools* est pensé pour exploiter des configurations matérielles puissantes afin de proposer des environnements virtuels très détaillés, *SGTools* est limité en terme de performances mais bénéficie d'un mode de diffusion large et multiplateforme par le biais d'Internet. Ciblant l'industrie du Serious Game, *SGTools* est également compatible avec la spécification *SCORM*, lui permettant d'être intégré à des plateformes *LMS* *(p.145)*.

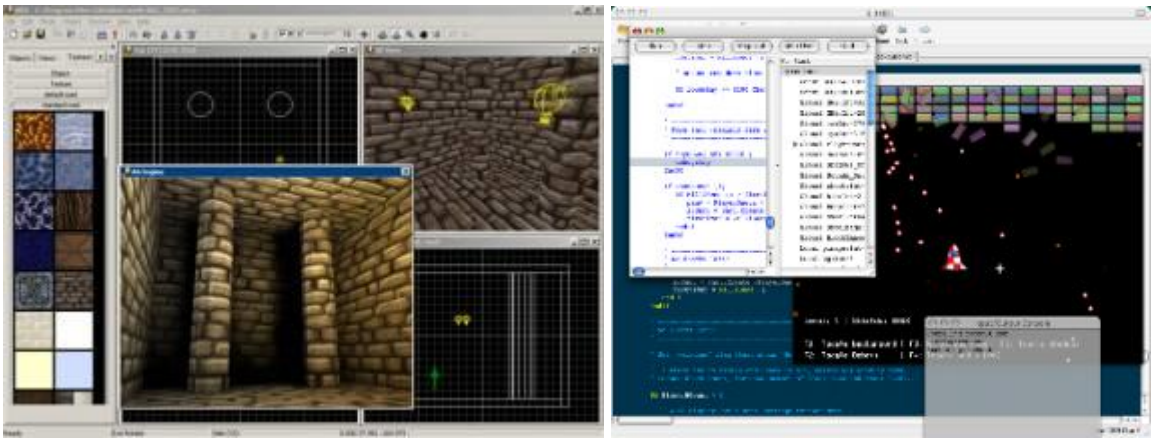
2.6 CLIQUER OU PROGRAMMER ?

Nous venons de présenter quelques logiciels qui permettent de créer tout type de jeu vidéo sans langage de programmation. Au total, 60 usines à jeux généralistes de notre corpus permettent de créer des règles de jeux sans avoir obligatoirement recours à un langage de programmation *(51 « éditeur visuel » et 9 « configurations de paramètres)*. Si ces approches semblent les plus pertinentes pour permettre aux novices de s'adonner à la création vidéoludique, elles ne sont pas les seules disponibles. Ainsi, 89 autres outils s'appuient obligatoirement sur un langage de script propriétaire *(47 outils)* et/ou un langage de programmation commun *(42 outils)* pour permettre aux utilisateurs de créer les règles de leurs

jeux. Si ces deux approches correspondent à des logiciels plus ou moins équivalents en terme de puissance, les usines à jeux basées sur un langage de programmation semblent permettre un passage plus aisé vers le monde professionnel, tels que l'illustrent les exemples ci-après.

2.7 DES LANGAGES DE SCRIPTS INSPIRES DES LANGAGES DE PROGRAMMATION

La plupart des usines à jeux uniquement basées sur l'écriture de code informatique s'appuient soit sur un langage de programmation commun, comme *Java*, *C++*, *C#*, *Lua* ou *BASIC*, soit sur un langage de script propriétaire relativement élaboré. Du côté de ces derniers, nous retrouvons par exemple la série des *3D Game Studio* ***(Conitec, 1993-2010)***. L'origine de cette usine à jeux remonte à un moteur 3D open-source baptisé *ACK-3D* ***(Larry Myers, 1993)***. Son code fut publié en 1993 dans l'ouvrage *Amazing 3-D Games Adventure Set: The Best Way to Create Fast Action 3-D Games in C*, écrit par ***Larry Myers***, également auteur de l'outil. Le code de ce moteur fut repris par ***Johann Christian Lotter***, qui l'améliora afin de créer *ACK NEXT GENERATION* en 1994. ***Lotter*** fondera ensuite la société ***Conitec*** pour continuer à faire évoluer son logiciel. Il le transformera en une célèbre usine à jeux utilisant une représentation 3D connue sous le nom *3D Game Studio*. Cet outil propose un éditeur de niveaux en 3D qui permet l'importation de nombreux formats de modèles 3D. Son langage de script propriétaire, le *Lite-C*, est une combinaison allégée du *C* et du *C++*.



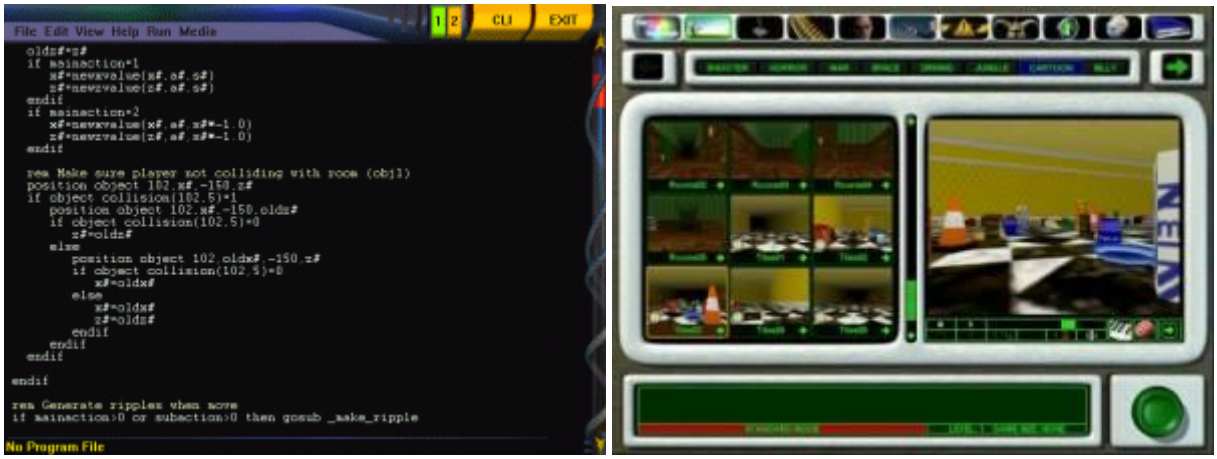
63. ***3D Game Studio A6*** ***(gauche)*** **et** ***BlitzMax*** ***(droite)***

Une telle histoire, dans laquelle un petit logiciel inventé par une personne seule finit par évoluer en usine à jeux commercialisée par une société privée, n'est pas unique dans notre corpus. Ainsi, la série des *Blitz* ***(Mark Sibly, 1993-2004)*** a débutée avec la création d'un dérivé du langage *BASIC* dédié à la création de jeux pour *Amiga* : le *Blitz Basic II* ***(Mark Sibly, 1993)***. Ce langage continuera à évoluer pour devenir la colonne vertébrale d'une famille d'usines à jeux créés et édités par la société ***Blitz Research***, fondée par ***Mark Sibly***. Cette famille regroupe *Blitz2D* ***(2000)*** et *BlitzPlus* ***(2003)***, qui permettent de créer des jeux en 2D, et *Blitz3D* ***(2001)*** et *BlitzMax* ***(2004)***, qui sont spécialisés dans la 3D.

2.8 THE GAME CREATORS, DU BASIC AU C++

Dans notre corpus, le langage *BASIC* est souvent utilisé comme base pour les langages de script propriétaires. Au-delà de la famille *Blitz* et de *3D Rad* ***(Studio Bitplane, 1998-2011)***, un autre exemple célèbre est le logiciel *DarkBasic* ***(The Game Creators, 1999)***. Il fut inventé par ***Lee Bamber*** et ***Rick Vanner***. Ces deux programmeurs fondèrent la société ***The Game Creators*** pour le commercialiser. A noter que ***Lee Bamber*** n'était pas un novice dans le monde des usines à jeux, puisqu'il a notamment créé les exemples de jeux accompagnant *The Games Factory* (***Clickteam, 1998)***. Complètement tourné vers la programmation, *DarkBasic* ne propose aucun éditeur visuel. La création de niveaux se fait donc en important des modèles

3D au travers de scripts écrits en *BASIC*. Le succès commercial de *DarkBasic* donna lieu à toute une série d'usine à jeux basées sur divers langages de programmation. *PureGDK* ***(Matthew D'Onofrio, 2008)*** se concentre sur le langage *PureBASIC*, alors que *DarkGDK* ***(The Game Creators, 2008)*** s'ouvre aux langages *Visual C++* et *C# .NET*.



64. *DarkBasic* *(gauche)* et *The 3D Gamemaker* *(droite)*

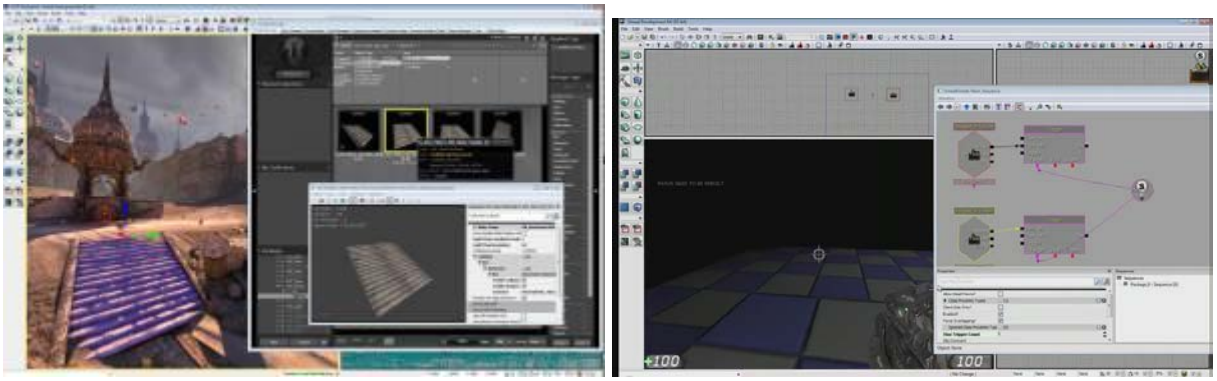
Tous ces logiciels rencontrent suffisamment de succès pour permettre à la société ***The Game Creators*** de se diversifier. Par exemple, elle rachète l'usine à jeux 3D *X-Quad Game Editor* ***(Odyssey Creators, 2007)***, qui propose uniquement un éditeur visuel au lieu d'un langage de programmation. La société fait également l'acquisition de *Realm Crafter* ***(Solstar Production, 2004),*** un outil spécialisé dans la création de jeux vidéo massivement multijoueurs en ligne *(MMO)*. Plus étonnant, en rachetant *Leadwerks Engine 2* ***(Leadwerks Software, 2008)***, elle propose un produit concurrent à son *DarkBasic*, bien qu'il soit situé dans une gamme de prix le destinant aux professionnels plutôt qu'aux amateurs. ***The Game Creators*** se développe également en créant d'autres outils en interne. *FPS Creator* ***(2005)*** et *FPS Creator X10* ***(2008)*** sont des usines à jeux spécialisées dans les jeux de tir en vue subjective. Elles permettent de créer rapidement des jeux de ce genre grâce à un éditeur de niveau 3D très simple d'accès. Une large bibliothèque de modèles 3D est alors disponible pour peupler ces niveaux. La modification des règles de jeux se fait en associant des scripts préprogrammés à ces modèles 3D, *FPS Creator* étant livré avec tous les éléments caractérisant ce genre vidéoludique. Mais le produit le plus étonnant du très large catalogue d'usine à jeux proposé par ***The Game Creators*** est sans doute *The 3D Gamemaker* ***(2001)***. S'il permet de créer très facilement des jeux en 3D, sa facilité d'utilisation extrême se fait au détriment de la créativité. Ainsi, il n'est plus possible de modifier soi-même les règles du jeu. La création se fait simplement en piochant des éléments dans une bibliothèque et en les positionnant sur une carte pour construire des niveaux. Il suffit ensuite de régler quelques paramètres de base *(vitesse de déplacement, nombre de vies…)* pour créer un jeu vidéo distribuable sous forme autonome. Signe que la créativité est ici limitée à une suite de choix basiques, ce logiciel propose un mode *« création automatique »*. Le logiciel génère alors un jeu par lui-même, en réglant les quelques paramètres proposés au hasard, et en agençant les éléments de manière aléatoire pour construire un niveau.

Au final, le cas de ***The Game Creators*** est particulièrement remarquable car, dans notre corpus, il s'agit de la seule société qui distribue une aussi grande variété d'usines à jeux. La plupart de ses concurrents semblent plutôt se focaliser sur une seule famille de logiciels.

2.9 USINES A JEUX OU MIDDLEWARE ?

Comme nous pouvons le remarquer, ces quelques exemples d'usines à jeux généralistes s'appuyant sur un langage de programmation sont commercialisés par de très petites

structures. Leurs logiciels sont donc situés dans des gammes de prix les rendant accessibles aux amateurs, voire à des professionnels de la scène indépendante *(p.47)*. En conséquence, la technologie proposée *(moteurs graphique, physique, IA…)* n'est pas comparable à celle utilisée dans l'industrie du jeu vidéo. Il existe pourtant des « usines à jeux » reposant sur des technologies de rendu 3D, de simulation physique ou d'intelligence artificielle à la pointe. Cependant, au vu de leur prix de vente, ces logiciels sont plutôt utilisés par les professionnels de l'industrie. Nous avons déjà évoqué le cas de *Virtools*. Nous pouvons également citer *Unreal Development Kit* ***(Epic, 2009)***, que nous avons déjà mentionné lors de l'évocation du *« middleware »* *(p.46)*. En effet, les « middlewares » utilisés par les professionnels ne sont finalement que des « usines à jeux » proposant une technologie à la pointe, assortie d'un prix de vente élevé. Il existe pourtant une différence d'approche entre les outils destinés aux amateurs et les « middlewares ». Les professionnels bénéficiant d'une solide formation technique *(programmation, modélisation 3D, graphisme…)*, leurs « usines à jeux » ne cherchent pas à limiter la puissance des outils pour les rendre plus accessibles. Elles visent plutôt à « pré-macher » les tâches répétitives du processus de développement de manière à accélérer au maximum la création de jeux industriels. Ainsi, bien que l'*Unreal Development Kit* soit dorénavant disponible gratuitement pour toute personne créant des jeux non-commerciaux, le savoir technique requis pour utiliser cette usine à jeux la met hors de portée de la plupart des amateurs. Il en va de même pour la majorité des autres « middleware », quel que soit leur prix de vente.



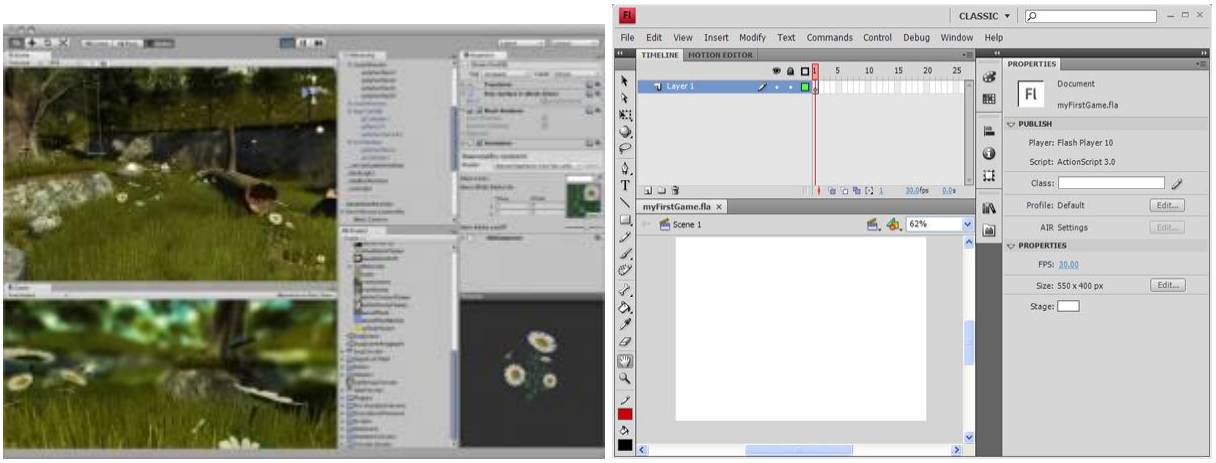
65. *Unreal Development Kit* : éditeur de niveaux *(gauche)* et de règles *(droite)*

2.10 DES LOGICIELS DE CREATION PROFESSIONNELS ACCESSIBLES AUX AMATEURS

Si, jusqu'à la fin des années 1990, les usines à jeux étaient principalement l'apanage des amateurs, aujourd'hui la frontière entre utilisateurs amateurs et professionnels est parfois floue. Ainsi, nous trouvons dans notre corpus des usines à jeux comme *Unity* ***(Unity Technologies, 2005-2010)***, et des logiciels de création multimédia tels que *Flash* ***(Macromedia, 1996-2010)*** et *Director* ***(Macromedia, 1988-2009)***. Ces outils sont originellement destinés aux professionnels. Leur prix de vente de plusieurs centaines d'euros confirme d'ailleurs cette orientation. Dans le cas de *Flash* et de *Director* il s'agit même d'outils qui ne sont pas originellement destinés à faciliter la création de jeux, mais plutôt de sites Internet et autres applications multimédia. Pourtant, 34% des Serious Games du corpus rassemblé sur le site *Serious Game Classification* *(p.32)* sont créés par des professionnels grâce la technologie *Flash (760 jeux sur 2218)*.

Cette popularité est sûrement liée au fait que *Flash* et *Director* s'appuie sur des approches de facilitation similaires à celles des « usines à jeux » destinées aux professionnels, comme *Unity*. Pourtant, la simplicité d'utilisation de ces trois outils « professionnels » est en tout point comparable à celles des usines à jeux de notre corpus qui sont utilisées par des amateurs. Tout d'abord, ces trois outils simplifient au maximum la création d'univers de jeux ou d'animations interactives grâce à des éditeurs visuel très intuitifs. Pour la création de

règles, ils s'appuient sur un langage de script propriétaire. Sa complexité est largement comparable aux langages de scripts des autres outils de notre corpus. Si, en théorie, ces logiciels sont réservés aux créateurs professionnels, dans la pratique ils sont également utilisés par une large frange de la sphère amateur. Est-ce du au fait que les amateurs aspirent parfois à devenir de futurs professionnels *(p.51)* ? Ou tout simplement à la circulation de versions pirates de ces trois outils, dont la simplicité d'utilisation permet à des amateurs de les utiliser facilement une fois que leur prix de vente n'est plus un obstacle ?



66. *Unity 2.6 (gauche)* **et** *Flash CS4 (droite)*

En réaction à ce « plébiscite » spontané *(et non anticipé)* de certains Game Designers amateurs, les sociétés éditrices de ces logiciels ont proposé une version gratuite *(et légale)* de leur outil. Elles sont accessibles à toute personne ne commercialisant pas ses réalisations. Par exemple, à partir de la version *Unity 2.6* ***(Unity Technologies, 2009)***, ce logiciel est gratuit pour une utilisation non commerciale. De son côté, la société ***Adobe***, qui édite dorénavant le logiciel *Flash*, distribue une version gratuite et (partiellement) open-source du compilateur à la base de sa technologie. La société continue bien évidement à commercialiser un logiciel avec de nombreux éditeurs permettant de créer du contenu graphique et d'écrire des scripts dans son langage de programmation propriétaire, *ActionScript*. Mais le compilateur pour ce langage étant dorénavant gratuit, des amateurs ont réalisé des outils de création alternatifs dédiés à la technologie *Flash*. Parmi ces initiatives, notre corpus recense deux outils clairement centrés sur la création de jeux vidéo : *Flixel* ***(Adam Saltsman, 2009)*** et *FlashPunk* ***(Chevy Ray Johnston, 2009)***. Couplés à des éditeurs de code *ActionScript* gratuits tels que *FlashDevelop* ***(FlashDevelop Team, 2005-2010)***, ces deux outils open-source simplifient grandement la création de jeux vidéo en *Flash*. Si l'apprentissage de l'*ActionScript* est toujours nécessaire, l'ajout de fonctions dédiées au jeu ainsi que d'éditeurs de niveaux visuels rend ces outils bien plus adaptés à la création vidéoludique que le seul compilateur de code rendu gratuit par ***Adobe***.

Les utilisateurs amateurs ayant ainsi accès de manière tout à fait officielle à des outils utilisés par les professionnels, ceux qui font l'effort d'apprendre à les utiliser bénéficient d'une passerelle vers le monde professionnel. Cela tend à rendre d'autant plus floue la frontière entre la sphère amateur et la sphère des professionnels indépendants. En utilisant ces logiciels, les amateurs apprennent à programmer avec des langages utilisés par l'industrie. Ils sont donc susceptibles de devenir des programmeurs professionnels. Nous retrouvons ici un phénomène comparable à ce que nous avions observé pour le *« modding »* *(p.51)*.

2.11 CREER DES JEUX AVEC LES LOGICIELS D'INITIATION A LA PROGRAMMATION

Abordons à présent une dernière catégorie d'outils utilisés par les Game Designers amateurs. D'une manière générale, la programmation informatique est un des métiers de la création

vidéoludique qui nécessite le savoir technique le plus conséquent. Il n'est donc pas étonnant de voir des amateurs créer des jeux vidéo en utilisant des logiciels facilitant l'apprentissage de la programmation. Ces logiciels simplifient grandement la création de tout type de programmes informatique. Par extension, ils simplifient donc la création de jeux vidéo. Les amateurs n'hésitent alors pas à utiliser ces outils comme des usines à jeux « généralistes », même s'il ne s'agit pas là de leur vocation première.

2.11.1 LES LANGAGES DE PROGRAMMATION SIMPLES

Les principales cibles de tels logiciels pédagogiques sont les enfants, ou tout autre personne novices souhaitant apprendre à programmer. Nous pensons en premier lieu au *LOGO* ***(Wally Feurzeig & Seymour Papert, 1967)***. Ce langage de programmation a été rendu célèbre par sa tortue, une sorte de curseur virtuel que les utilisateurs peuvent manipuler avec des instructions simples afin de dessiner des formes géométriques. Signe de l'ingéniosité de cette approche, des tortues robotisées et programmables en *LOGO* ont même été construites. L'impact du *LOGO* sur la recherche en pédagogie est conséquent, notamment pour l'enseignement touchant à l'informatique (Papert, 1993). De nos jours, des chercheurs continuent à explorer les concepts posés par ***Seymour Papert*** pour travailler avec les enfants, à l'image du projet *POGO* *(p.236)*. Cette influence est également perceptible dans l'univers des usines à jeux avec un logiciel comme *Kodu Game Lab* ***(Microsoft, 2009)***. Il propose de créer des jeux 3D en programmant des avatars virtuels dotés de capacités prédéfinies, qui ne sont pas sans rappeler celles de la *tortue LOGO*. Au lieu d'avoir recours à un langage de programmation, *Kodu Game Lab* propose de programmer les objets avec une approche par « blocs » qui rappelle celle de *Game Maker*. L'utilisateur programme chaque objet en créant des règles composées de *« conditions »* et *« d'actions »*, chaque objet du logiciel ayant un ensemble de *« conditions »* et *« d'actions »* qui lui sont spécifiques *(certains objets peuvent voler, d'autres rouler, certains peuvent tirer, suivre un chemin…)*. Cette approche a même permis d'élaborer un nouveau langage de programmation d'avatars (Stolee, 2010). Enfin, le langage *LOGO* a également été utilisé en contexte scolaire, pour des ateliers des enfants en école primaire se voyaient proposer de réaliser des Serious Games pour leurs camarades. Nous reviendrons sur cette expérimentation menée par ***Kafai*** dans la dernière partie de la thèse *(p.237)*.

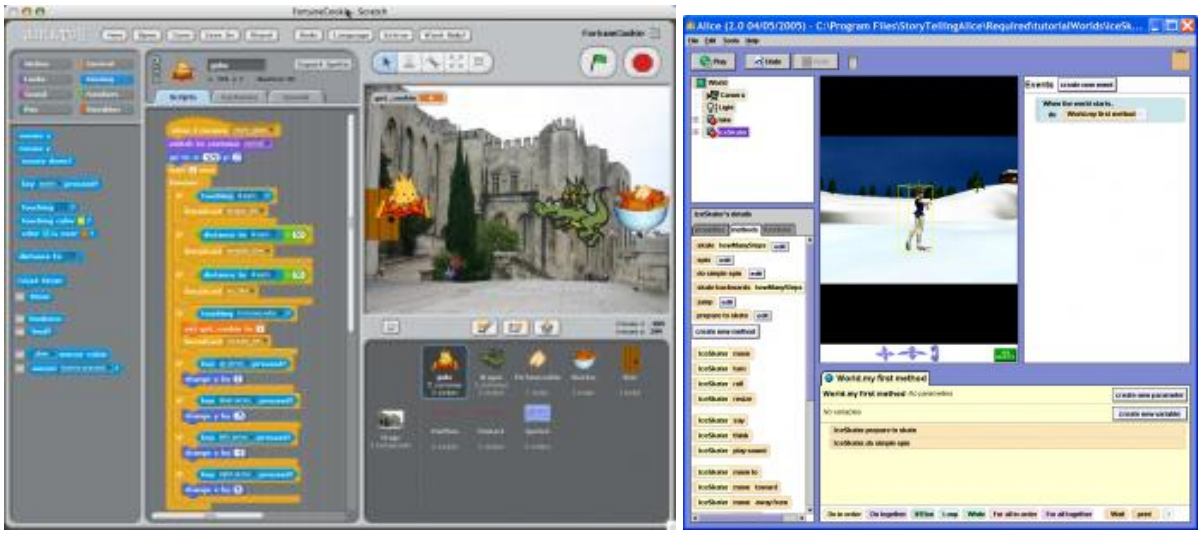
67. *Kodu Game Lab* *(gauche)* et *Kid's Programming Language* *(droite)*

Dans une veine similaire, d'autres outils incitent les utilisateurs à apprendre un langage de programmation afin de créer des jeux vidéo. En plus des incarnations modernisées du *LOGO*, telle que *StarLogo*, nous identifions les logiciels *Kid's Programming Language* ***(Jonah Stager, 2005)***, *Phrogram* ***(The Phrogram Company, 2006)*** ou encore *Tangara* ***(Colombbus, 2009)***. Globalement, ces langages de programmation sont assez proches des langages de scripts proposés par les usines à jeux généralistes. En plus des fonctionnalités générales de programmation *(variables, structures de contrôle…)*, elles proposent des fonctions dédiés à la

création de jeux vidéo : *manipulation d'éléments graphiques, saisies des interactions, moteurs de mouvements préprogrammés, gestions de collisions…*

2.11.2 LES APPROCHES GRAPHIQUES DE LA PROGRAMMATION

Dans un autre registre, nous pouvons penser au logiciel *Scratch* ***(MIT Media Lab, 2007)***, qui permet de créer des programmes par l'associations d'instructions représentées par des « blocs » que l'on assemble. Cette logique découle du langage de programmation *Squeak* ***(Alan Kay & Dan Ingalls & Adele Goldberg, 1996)***, dont *Scratch* est une version améliorée. L'approche graphique de la programmation proposée par *Scratch* rencontre un succès considérable, notamment auprès des enseignants qui l'utilisent avec leurs élèves. En ce sens, il est un peu la version moderne du *LOGO*. Mais, présence d'Internet oblige, ce succès est aujourd'hui centralisé autour d'une communauté d'utilisateurs. Les créateurs du logiciel ont développé une plateforme de partage des créations réalisées avec *Scratch*. Nous avons déjà évoqué l'importance des communautés d'utilisateurs en ligne pour le succès des usines à jeux *(p.170)*. Visiblement, cela est également applicable à *Scratch*. Pas moins de 1.612.511 projets[58], qui peuvent être des animations interactives comme des jeux vidéo, ont été partagés sur le site officiel de l'outil. De plus, la nature open-source de *Scratch* lui permet d'évoluer dans plusieurs directions simultanées. Ainsi, si le groupe original de développement focalise ses efforts sur l'aspect « partage » pour transformer *Scratch* en avatar du « Jeu 2.0 » *(p.200)*, d'autres projets lui ajoutent des fonctionnalités de programmation supplémentaires. Un des plus notable est le projet *Build Your Own Blocs* ***(Jens Mönig, 2009)***, qui permet à l'utilisateur de créer ses propres blocs d'instructions en combinant des blocs de bases. Nous retrouvons là une fonctionnalité évoquée en détail pour *Virtools* *(p.182)*. Mentionnons également *Panther* ***(Sparks & al., 2010)***, qui ajoute à *Scratch* de nombreux blocs d'instructions destinés à faciliter la création de jeux vidéo.



68. *Scratch* *(gauche)* et *Alice* *(droite)*

Ce genre d'approche graphique de la programmation se retrouve d'ailleurs dans d'autres logiciels. Certains outils proposent même d'éluder complètement les instructions de programmation, pour aborder le concept d'algorithmique avec des symboles entièrement graphiques. Par exemple, la série des *Baltie* ***(SGP Systems, 1996-1999)*** permet de créer des programmes impliquant un avatar en agençant des cartes représentant les différents mouvements du personnage. Dans une veine similaire, *ToonTalk* ***(Ken Kahn, 1998-2007)*** permet de créer des programmes en déplaçant des objets et en répondant à des *QCM*. De son côté, *Alice* ***(Randy Pausch & al., 1995-2010)*** utilise une approche graphique de la programmation ressemblant beaucoup à celle de *Scratch*. Si *Scratch* est limité à de la 2D,

[58] Relevé le 25-02-11 à partir de http://stats.scratch.mit.edu/community/

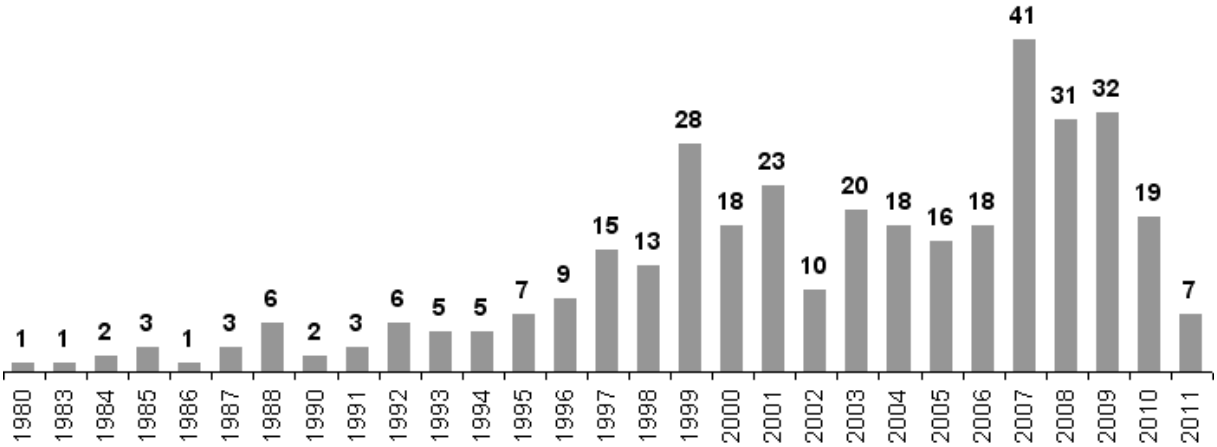
Alice permet la création d'histoires interactives, de simulations en réalité virtuelle et de jeux vidéo en 3D temps réel. Afin de faciliter la création de telles applications, ce logiciel est livré avec de nombreux modèles d'objets et personnages. Il permet ainsi de créer une application tout en évitant la longue et complexe phase de modélisation d'objets 3D. Inventé par un groupe de chercheurs dirigé par ***Randy Pausch***, le premier objectif visé par *Alice* était de proposer un système de prototypage rapide pour la création d'applications de réalité virtuelle (Pausch et al., 1995). Cette première version impliquait la programmation de scripts en *Python*. Cependant, le logiciel a rapidement évolué vers une interface graphique permettant de créer des programmes uniquement par l'association de « blocs d'instructions ». La base d'utilisateurs du logiciel en fut alors considérablement élargie, puisque les dernières versions d'*Alice* sont utilisées pour l'apprentissage de l'informatique. Une étude sur la pertinence du logiciel contexte pédagogique a d'ailleurs été menée pour évaluer la valeur ajoutée d'*Alice* dans des cours d'introduction à l'informatique dispensés à des lycéens (Moskal, Lurie, & Cooper, 2004). L'étude révèle que si l'utilisation du logiciel n'apporte pas de gains significatifs pour une population « aisée » *(accès à l'informatique à la maison…)*, elle permet en revanche de combler l'écart de performance entre les élèves de la catégorie « défavorisée » et les autres. En d'autres termes, selon cette étude, l'utilisation d'un tel logiciel en contexte pédagogique semble une piste intéressante pour tenter de lutter contre la « fracture numérique » liée à la différence de milieu social entre les élèves. S'il ne s'agit que d'une expérimentation isolée, cet exemple nous invite à réfléchir plus largement sur le potentiel des outils facilitant la création de jeux vidéo pour mettre en place des activités pédagogiques. Nous reviendrons sur cette question lors de la dernière partie de cette thèse *(p.234)*.

Au final, s'ils sont avant tout destinés à faciliter l'apprentissage de la programmation, tous ces logiciels peuvent également servir à créer des jeux, comme l'ont compris certains Game Designers amateurs. A l'inverse, nous observons que les « usines à jeux » sont parfois un excellent tremplin pour l'apprentissage de la programmation informatique (Overmars, 2004).

3 DONNEES QUANTITATIVES POUR UN LARGE CORPUS D'USINES A JEUX

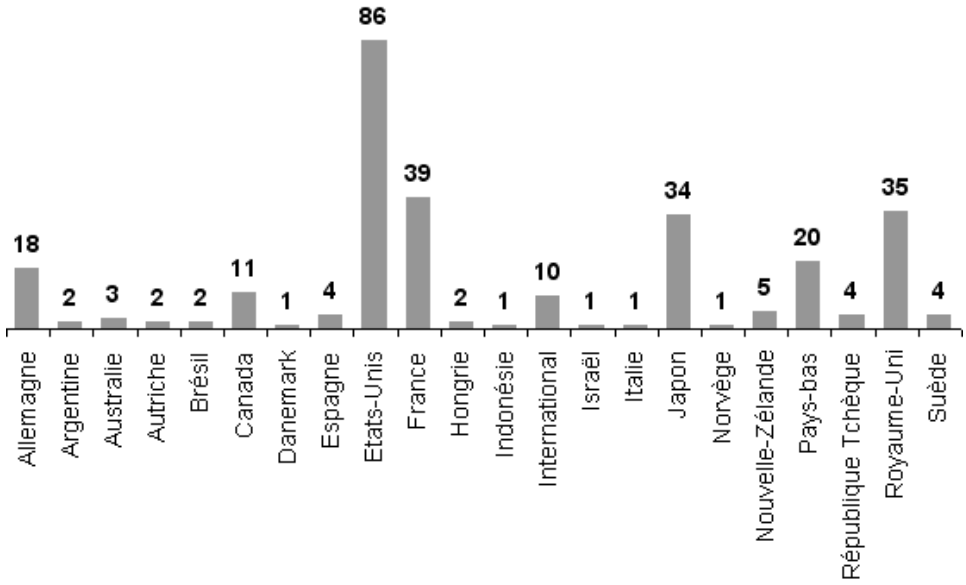
Notre corpus de 363 usines à jeux *(ou logiciel assimilé)* est malheureusement trop volumineux pour être intégralement détaillé dans cette thèse. Après avoir présenté ces quelques exemples qualitatifs, intéressons-nous à présent aux données quantitatives issues de l'ensemble de ce corpus.

3.1 DONNEES GENERALES



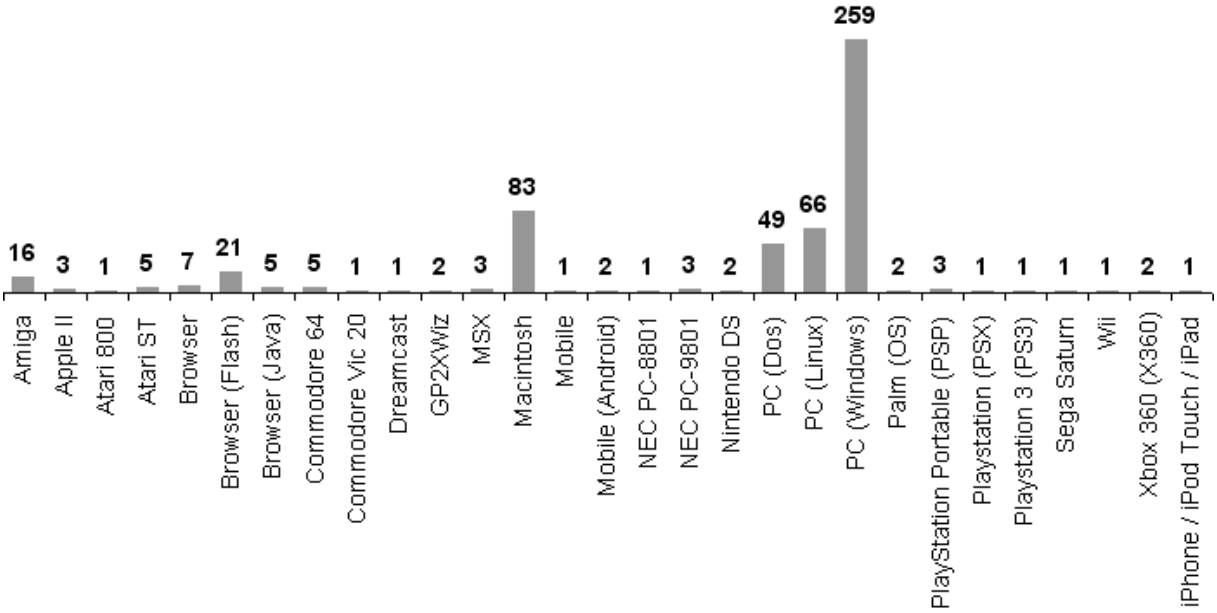
69. Années de sortie des usines à jeux de notre corpus

Pour commencer, nous observons une croissance globale du nombre d'usines à jeux publiées au fur et à mesure des années. Deux pics semblent néanmoins se dégager. Le premier, entre les années 1999 et 2000, est apparemment lié à deux facteurs. Tout d'abord, la démocratisation d'Internet auprès du grand public qui facilite la distribution de tels outils. Ensuite, l'apparition des premières versions d'usines à jeux qui dominent aujourd'hui. Il y a d'un côté des outils destinées à la 3D (*NeMo*, *DarkBasic*, *Alice 2.0*…) qui était jusqu'alors plutôt rares, et de l'autre des outils 2D (*Game Maker*, *Multimedia Fusion*, *RPG Maker 2000*, *M.U.G.E.N.*…) qui améliorent les bases posées par leurs prédécesseurs. Le second pic observé à l'année 2007 semble tout simplement correspondre à l'arrivée du « Jeu 2.0 » *(p.200)*, dont les principaux représentants sont sortis entre les années 2007 et 2008.



70. Pays d'origine des concepteurs des usines à jeux de notre corpus

En ce qui concerne les pays d'origine des développeurs d'usines à jeux, nous n'avons pas été en mesure de déterminer cette donnée pour 77 outils. Au sein des 286 usines à jeux correctement identifiées, les Etats-Unis dominent clairement les autres pays : ils représentent pratiquement un tiers des outils produits. La France, le Royaume-Uni et le Japon suivent avec une part située entre 12 et 14%. L'Allemagne et les Pays-Bas avoisinent les 5%, alors que les autres pays de la liste ne représentent que quelques usines à jeux de notre corpus. A noter d'ailleurs que seuls 21 pays du monde sont ici représentés, plus une catégorie *« International »* qui recense les outils développés par des personnes issues de pays différents *(souvent le cas de projets open-source)*. Au final, l'Europe représente 131 usines à jeux, soit pratiquement 46% du corpus, alors que le continent Américain est à 35% avec 101 outils, et le Japon à seulement 11%. En regard des marchés du jeu vidéo de divertissement, la part du Japon est surprenante étant donné qu'il s'agit d'un marché historiquement dominant, en particulier par rapport à l'Europe (Kohler, 2004). Elle s'explique néanmoins par la prédominance des jeux sur support console au Japon, alors que l'Europe se caractérise plutôt par une large part d'ordinateurs personnels. Or, les usines à jeux se trouvent principalement sur ordinateurs, et très rarement sur console, comme l'illustre le graphique ci-dessous :

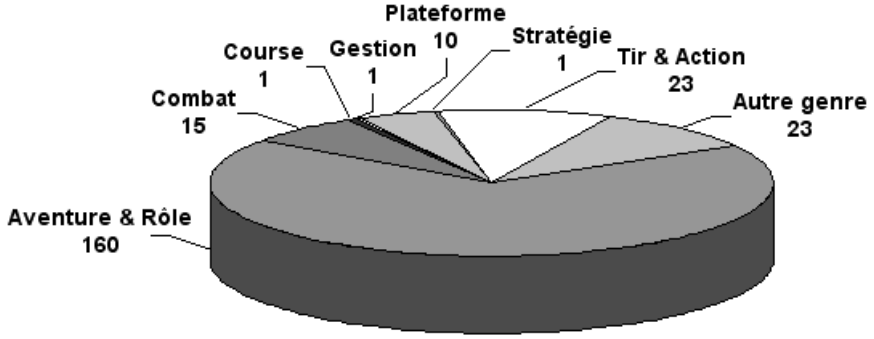


71. Supports des usines à jeux de notre corpus

Nous observons ici que la part du *PC (tout systèmes d'exploitation confondus)* dépasse largement les autres plateformes, à l'exception du *Macintosh*. Au final, tout en prenant en compte le fait que de nombreuses usines à jeux sont disponibles sur plusieurs supports, les 363 outils de notre corpus sont destinés à 90% aux ordinateurs personnels et seulement à 2% aux consoles. Les 8% restants concernent les usines à jeux disponibles directement dans un navigateur Internet (6%) et celles pour téléphones mobiles (2%). A l'image de la répartition des plateformes, les modes de distribution sont également très polarisés. 308 outils (85%) s'appuient sur une distribution dématérialisée *(Internet, BBS…)*, contre seulement 52 usines à jeux disponibles en « boîte » (15%). De même, 222 usines à jeux sont gratuites (61%) pour 168 logiciels disponibles en version payante (46%), certains cumulant les deux statuts avec une version « lite » gratuite et une version payante plus complète. Notons enfin que 67 usines à jeux (14%) sont des logiciels libres sous licence « open-source ».

3.2 OUTIL GENERALISTE OU OUTIL SPECIALISE ?

Dans notre corpus, nous observons **136 « usines à jeux généralistes »** *(p.177)* soit 37% des outils étudiés. Les 63% restants regroupent les **227 « usines à jeux spécialisées »** *(p.166)* dans un ou plusieurs genres de jeux vidéo, répartis comme indiqué ci-dessous :

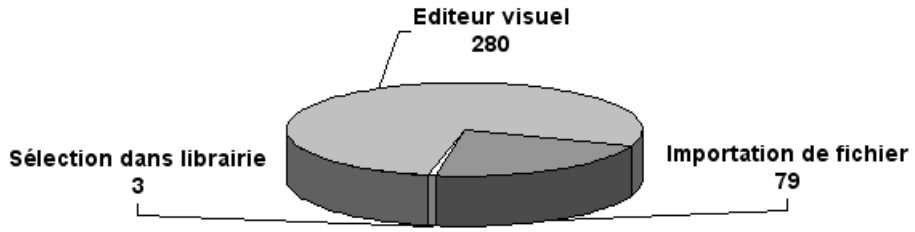


72. Répartition des genres de jeux vidéo des 227 usines à jeux spécialisées de notre corpus

Au-delà de ces données générales, nous pouvons également analyser les statistiques relatives à chacune des quatre catégories du *modèle ISICO étendu (p.160)*, afin d'identifier d'éventuelles méthodes de création dominantes.

3.3 INITIAL STATE

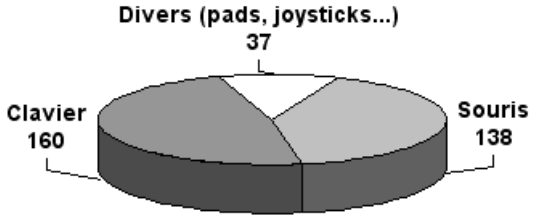
Seules 2 usines à jeux de notre corpus ne permettent pas de créer *« l'état initial »* du jeu. Pour les 361 logiciels restants, nous pouvons clairement observer que la méthode *« Editeur visuel »* est la plus courante, et fait écho à la popularité des « éditeurs de niveaux » *(p.50)*. Les 79 outils proposant l'importation de fichier sont, pour la plupart, des outils spécialisés dans la création de fictions interactives dépourvus d'éditeur de texte intégré *(p.166)*, ainsi que quelques rares outils isolés ne proposant pas d'éditeur de niveau, comme *DarkBasic (p.184)*. Quant à la méthode de sélection de niveaux préconstruits, elle est clairement minoritaire, et par ailleurs globalement circonscrite au courant du « Jeu 2.0 » *(p.202)*.



73. Répartition des méthodes de création de « l'Initial State » dans notre corpus

3.4 INPUT

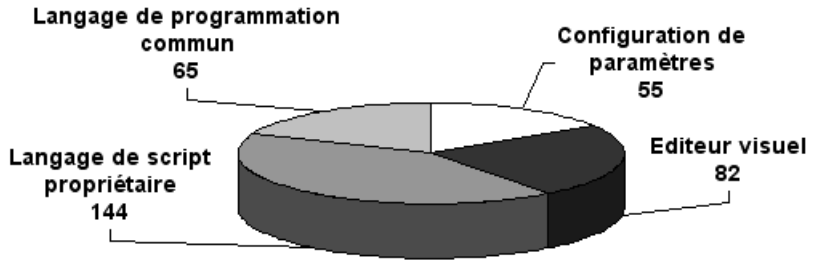
Comme nous l'avons déjà évoqué, de nombreux jeux proposent au joueur de configurer par lui-même l'interface entrante du jeu *(p.53)*. C'est également le cas des jeux créés avec les usines à jeux. Dans ce cas, elles ne permettent pas forcément au concepteur du jeu de la modifier. Il existe également de nombreux outils qui ne permettent tout simplement pas de créer directement l'interface entrante, celle-ci étant déjà « préconstruite » afin de faciliter la création, comme c'est parfois le cas des usines à jeux spécialisées *(p.166)*. Ainsi, seules 165 usines à jeux de notre corpus permettent au concepteur d'imaginer par lui-même les actions que déclencheront l'interface entrante, soit 45% des outils étudiés. Ce groupe d'usines à jeux permet la plupart du temps d'utiliser un clavier, une souris, et plus exceptionnellement d'autres périphériques :



74. Répartition des méthodes de création de « l'Input » dans notre corpus

3.5 COMPUTE

Les *« règles de jeu »* sont probablement la partie la plus complexe d'un jeu vidéo. Quatre méthodes très différentes existent pour les créer. Tout d'abord, la *« configuration de paramètres »* propose des choix limités et prédéfinis permettant de configurer certaines règles de jeu *(p.160)*. Ensuite, les *« langages de script propriétaires »* et les *« langages de programmation communs »* s'appuient sur l'apprentissage d'un langage de programmation pour écrire les règles de jeu sous forme de code informatique *(p.183)*. Enfin, les *« éditeurs visuels »* offrent une liberté de création équivalente aux langages de programmation, mais remplacent la saisie de code par la manipulation d'éléments graphiques, qu'il s'agisse « d'instructions » *(p.177)*, de « tableaux » *(p.178)* ou encore de « blocs » *(p.179)*. Ces quatre méthodes semblent avoir une répartition relativement homogène dans les 298 usines à jeux de notre corpus qui permettent de créer la partie *« Compute »* :

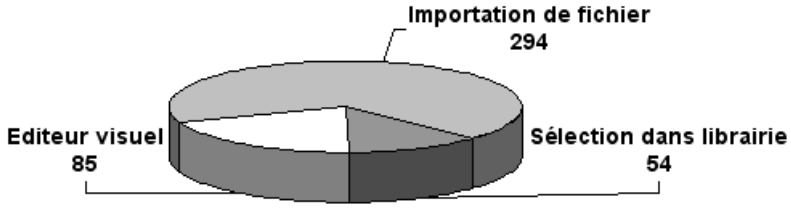


75. Répartition des méthodes de création du « Compute » dans notre corpus

A noter que certaines usines à jeux proposent plusieurs de ces méthodes pour la création des règles de jeu, à l'image de *Game Maker* *(p.179)* et de *Virtools* *(p.182)*.

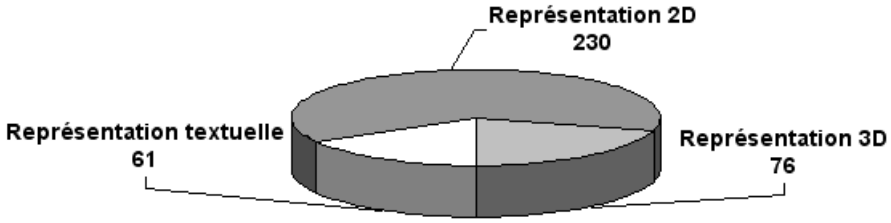
3.6 OUTPUT

Enfin, 354 usines à jeux permettent de modifier l'interface sortante du jeu avec une ou plusieurs méthodes. Comme nous pouvons le voir ci-dessous, la possibilité d'importer des fichiers *(images, sons…)* est largement répandue (83%). Un nombre non négligeable d'outils (24%) permettent également de créer directement des éléments sonore ou graphique par un *« éditeur de contenu émergeant »* dédié, n'obligeant donc pas l'utilisateur à recourir à des outils externes pour les créer. Enfin, 15% des usines à jeux sont livrées avec une librairie d'images et/ou de sons, accélérant d'autant plus la création de l'interface sortante :



76. Répartition des méthodes de création de « l'Output » dans notre corpus

Qu'elles permettent de la modifier ou non, nos 363 usines à jeux servent à créer des jeux possédant une interface sortante, comme tout jeu vidéo. Cette dernière peut être basée sur du texte (17%), sur des éléments graphiques en 2D (62%), ou sur de la 3D temps réel (21%) :



77. Répartition des modes de représentation des jeux créables avec les usines à jeux de notre corpus

4 RETOUR D'EXPERIENCE : CREER UN SERIOUS GAME AVEC UNE USINE A JEUX

Après avoir passé en revue les outils techniques de notre corpus permettant la création de nouveaux jeux vidéo autonomes, nous pouvons nous interroger sur leur potentiel pour la création de Serious Games. Nous avons déjà identifié quelques expériences d'utilisation de la création vidéoludique à des fins utilitaires, en général comme support à une activité pédagogique. Par exemple, *Quandary* a été utilisé pour travailler l'écriture et le français *(p.166)*, *Stagecast Creator* pour l'initiation à l'informatique *(p.180)*, et *Game Maker* pour l'apprentissage de la programmation *(p.179)*. Ces trois expérimentations utilisent des usines à jeux à des fins « sérieuses », mais elles s'appuient uniquement sur la création de jeux vidéo de divertissement. Est-il alors possible d'utiliser les usines à jeux pour créer des Serious Games ?

Afin de répondre à cette question, nous avons conduit une petite expérimentation sur l'utilisation d'une « usine à jeux » pour la création d'un Serious Game. Le contexte est ici quelque peu singulier. Avec les chercheurs ***Julian Alvarez*** et ***Olivier Rampnoux***, nous avons fondé l'association ***Ludoscience***, dont la mission est d'étudier le jeu vidéo sous toutes ses formes, et plus particulièrement celle du Serious Game. Ce travail nous a amené à écrire l'ouvrage de vulgarisation *Introduction au Serious Game* (J. Alvarez & Djaouti, 2010). En tant qu'association travaillant notamment sur le Game Design, nous avons accueilli en 2011 un jeune stagiaire souhaitant découvrir la création de jeux vidéo. Sa durée de stage étant très courte, nous lui avons alors proposé de réaliser un Serious Game visant une finalité très simple : faire la promotion de notre ouvrage. Nous avions besoin d'un sujet qui ne soit pas trop ambitieux en regard de la durée du stage, tout en permettant au stagiaire de s'impliquer totalement dans le processus de création. Un Serious Game à vocation publicitaire semblait donc tout à fait adapté, à plus forte raison pour mettre en avant un ouvrage traitant de Serious Games. Une fois le sujet établi, la question des moyens techniques fut à déterminer. L'objectif était que notre stagiaire, ***Quentin Vialade***, puisse réaliser lui-même des prototypes. Lors de notre premier entretien, ***Quentin*** nous exposa brièvement son profil. Etudiant en première année d'*IUT SéRéCom*, il n'avait jamais créé de jeux vidéo, ne maîtrisait pour l'instant aucun langage de programmation, n'avait pas de compétences particulières en infographie et ne disposait que de trois semaines de stage. De plus, notre association ne possède pas de locaux et fonctionne principalement sur le mode du télétravail. Il était donc indispensable que ***Quentin*** puisse avoir une relative autonomie technique, afin de pouvoir véritablement se concentrer sur la création vidéoludique. Pour l'aspect théorique du *« Serious Game Design »*, nous avons décidé de lui assurer une formation continue par la pratique. Seule restait la question de l'outil technique à utiliser pour la réalisation de prototypes. Nous avons alors étudié notre corpus d'outils techniques pour identifier celui qui semblerait le plus pertinent par rapport aux contraintes évoquées. Notre choix s'est porté sur *The Games Factory 2 (p.178)*. Il s'agit d'une usine généraliste, laissant donc à ***Quentin*** une grande liberté de création. Le fait qu'elle soit limitée à la création de jeux en 2D simplifie grandement le travail graphique, le logiciel proposant un éditeur de dessin intégré. La dimension publicitaire du jeu implique une diffusion large auprès du grand public, qui renvoie à un mode de diffusion du jeu par le navigateur Internet. La technologie *Flash (p.186)* est à ce jour la plus adaptée à la diffusion de jeux par ce biais. Comme il n'était pas envisageable de former ***Quentin*** à *Flash* et à son langage de script propriétaire en si peu de temps, le fait que *The Games Factory 2* puisse exporter des jeux dans ce format répondait également à nos attentes.

Nous avons donc formé ***Quentin*** à l'utilisation de *The Games Factory 2* en environ deux heures, puis l'avons laissé s'approprier le logiciel au fur et à mesure qu'il travaillait sur le projet. Lors de ses trois semaines de stage, il nous a adressé quelques questions techniques liées à cet outil. La plupart du temps, il s'interrogeait sur la manière d'implémenter une fonctionnalité qu'il avait imaginée pour le jeu, par exemple *« comment faire rebondir mon*

personnage ? » ou *« comment faire pour créer un compte à rebours limitant le temps imparti pour finir le niveau ?»*. Il y a bien eu quelques questions liées à l'utilisation du logiciel. Mais globalement, la session de découverte en début de stage semble avoir été suffisante pour que l'apprentissage de la manipulation de cet outil ne soit pas un frein à sa créativité :

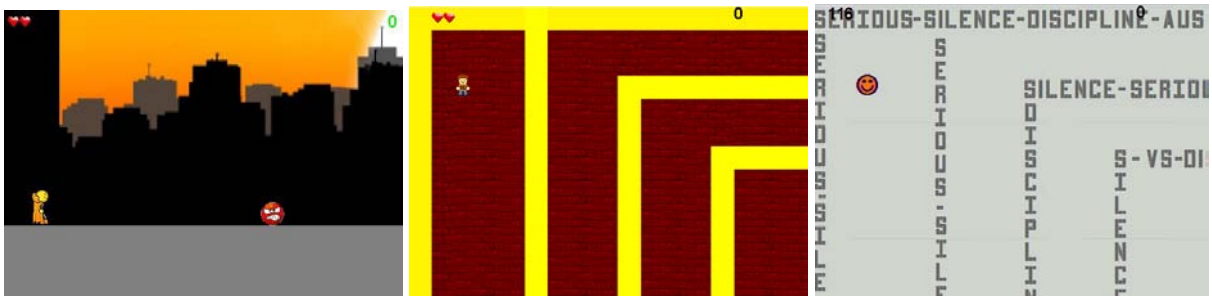
> *The Games Factory 2 est très intuitif, simple d'utilisation et laisse une grande place à l'innovation : on n'est pas obligé de se cantonner à deux ou trois styles de jeu. Le seul détail qui m'a gêné est lorsque je m'occupais du remplacement des murs par les mots : faire des sélections de groupes d'objets n'était pas du tout précis et cela me faisait perdre un peu de temps. Mais sinon, cet outil n'a pas été une limite à ma créativité. J'ai pu réaliser tout ce que je voulais : le double saut, des ennemis qui se déplacent verticalement et horizontalement…*

En utilisant ce logiciel, ***Quentin*** a donc pu s'immerger dans le processus de réalisation professionnel d'un petit Serious Game à vocation publicitaire. Voici globalement le cheminement qu'il a suivi tout au long de ces trois semaines, tel que raconté dans son rapport de stage (Vialade, 2011).

Tout d'abord, une phase de « réflexion papier » a permis d'étudier plusieurs concepts de jeux. La lecture de notre ouvrage et sa propre culture vidéoludique ont inspiré à ***Quentin*** l'idée d'un jeu dans lequel le joueur incarne un personnage ayant pour mission de faire découvrir le « Serious Game » à d'autres personnes. Son concept s'est ensuite affiné vers l'idée d'avoir des personnes « piégées » dans un niveau, le joueur incarnant alors un héros devant les délivrer en leur « apportant la connaissance ». Partant de là, ***Quentin*** a réalisé deux prototypes sous *The Games Factory 2* afin de trouver un genre vidéoludique adapté à cette idée de départ. Le premier prototype aborde une vue de côté et un gameplay du genre « plateforme », tandis que le second opte pour une vue de dessus avec un principe de jeu basé sur l'exploration de labyrinthes. Ces prototypes ont été testés par des amis de ***Quentin***, ainsi que deux *« âmes objectives et innocentes »* choisies par mon collègue ***Julian*** :

> *J'ai laissé mes enfants devant le jeu « vue de dessus » et cela fait plus de vingt minutes qu'ils sont dessus. Ils le recommencent sans arrêt. C'est bon signe car ils veulent le finir, alors que le jeu « vue de côté » ne les a pas du tout accroché.*

Nous avons donc opté pour le jeu « vue de dessus », pour lequel ***Quentin*** a développé un prototype plus élaboré. Il s'agit maintenant d'explorer le labyrinthe en évitant des ennemis et des pièges, à la recherche de clés de couleurs ouvrant les portes des cellules des prisonniers. La contrainte du jeu est incarnée par un temps limité pour finir le niveau. Afin de l'initier au *« Level Design »* *(p.46)*, nous avons demandé à ***Quentin*** de développer quatre niveaux de jeu sur ce principe. Le jeu commençait à prendre forme, mais il y avait néanmoins un problème de cohérence entre l'univers du jeu et le thème de notre livre. Après une phase de réflexion, un troisième prototype réalisé par ***Quentin*** modifie l'univers du jeu : les labyrinthes représentent désormais un univers froid pour illustrer le côté « serious » alors que le joueur incarne le « game » devant apporter une certaine chaleur. Les règles du jeu ont également été modifiées. Le système de vie a par exemple été abandonné au profit du seul principe de « compte à rebours », avec l'ajout de bonus accordant du temps supplémentaire. Ce processus de conception itératif a été poursuivi de manière à affiner le *« Level Design »* du jeu, afin qu'il délivre une expérience de jeu reposant sur un challenge progressif *(p.149)*. L'objectif est que le jeu captive suffisamment le joueur pour lui procurer une sensation de *« flow »* *(p.108)*.



78. Trois des prototypes réalisés par *Quentin* **sous** ***The Games Factory 2***

En conclusion de ce stage, nous avons demandé à ***Quentin*** ce qu'il avait pensé de cette petite expérience de Game Design, et plus particulièrement s'il aurait préféré réaliser un jeu vidéo destiné au seul divertissement plutôt qu'un Serious Game :

> *Je trouve que pour une première expérience de création de jeu vidéo, avoir créé un Serious Game est plutôt un avantage car dans ce genre de jeu tout l'univers est encore plus lié, connoté et logique que dans un jeu « normal ». Cela m'a donc permis de plus « garder les pieds sur terre » que si j'avais réalisé un jeu « normal ». Même si dans tous les cas j'aurais eu des limites car le jeu que j'aurais créé n'aurait pas été uniquement de mon cru. Par exemple, j'ai maintenant décidé de me lancer dans la création d'un « beat'em'all » à partir de The Games Factory 2, donc si j'ai envie de mettre un robot géant dans mon jeu, je peux. Alors que je n'aurais pas pu le faire dans le Serious Game réalisé pendant ce stage : que viendrait faire un robot géant dans nos labyrinthes ?*

De cette expérience nous retiendrons principalement deux éléments. Tout d'abord, et contrairement à ce que l'on pourrait penser, la création de Serious Games n'est pas forcément moins attirante que la création de jeu vidéo pour des joueurs passionnés. D'après le retour de ***Quentin***, nous pourrions même supposer que la création de Serious Games est plus adaptée à un cadre institutionnel, qu'il s'agisse du domaine de l'entreprise ou d'un contexte pédagogique. Mais surtout, les usines à jeux semblent être un moyen de permettre à des novices d'exprimer leur créativité. Ainsi, alors qu'il n'avait jamais réalisé de jeu vidéo de sa vie malgré son intérêt pour le domaine, ***Quentin*** a décidé de se lancer dans la réalisation d'un jeu vidéo en amateur avec *The Games Factory 2* suite à ce stage. Il lui a donc fallu attendre de trouver un outil technique à sa portée pour qu'il puisse réaliser son premier jeu vidéo. C'est en cela que la facilitation du processus de création vidéoludique apportée par les usines à jeux est la plus intéressante : **elle rend la création de jeux vidéo et de Serious Games accessible à des novices**. Mais *The Games Factory 2* n'est pas le seul logiciel à posséder un tel potentiel, comme nous avons pu l'observer à travers ce chapitre.

5 SYNTHESE : APPROCHES DE FACILITATION LIEES AUX USINES A JEUX

Si nous considérons l'ensemble des données quantitatives et qualitatives exposées dans ce chapitre, nous pouvons observer que le champ des usines à jeux rassemble une grande diversité de logiciels. Ceux-ci se différencient tout d'abord par des choix techniques, tels que ceux ayant traits au mode de représentation des jeux réalisables *(rendu textuel, à base de graphismes 2D ou de 3D temps réel)*. Les concepteurs d'usines à jeux opèrent ensuite des choix d'ordre philosophique, qui impactent directement le fonctionnement du logiciel et influent donc sur la démarche créative de ses utilisateurs :

- **Usines à jeux spécialisée ou généraliste ?** Ce choix permet d'adopter différentes approches de simplification. En se spécialisant dans un genre de jeu donné, les usines à jeux peuvent proposer de nombreuses parties « préconstruites » qui réduisent considérablement le temps de réalisation d'un jeu vidéo *(p.176)*. Mais cette approche peut parfois limiter la créativité des utilisateurs, et n'est de toute façon pas envisageable pour des outils généralistes. Ces derniers misent plutôt sur l'accessibilité de leur interface pour simplifier la création vidéoludique *(p.177)*.

- **Utilisation d'éditeurs de contenu émergeants ou de menus de choix prédéfinis ?** Les menus de choix prédéfinis sont plus simples d'utilisations, mais ils limitent la créativité aux bornes de l'imagination du concepteur de l'outil *(p.184)*. À l'inverse, les éditeurs émergeants permettent de créer du contenu qui n'aura pas forcément été anticipé par le concepteur de l'outil, au prix d'une prise en main légèrement plus complexe *(p.168)*. Chaque outil technique utilise une combinaison des deux approches qui lui est propre. Ce choix influe directement l'équilibre entre la liberté de création et la simplicité d'utilisation de chaque logiciel *(p.160)*.

- **La structure du jeu est-elle de type « objets intelligents » ou « objets conteneurs de données » ?** Tous les logiciels permettant de créer des jeux vidéo s'appuient sur une formalisation de ce qu'est un « jeu vidéo ». Comme nous l'avons vu avec les outils théoriques, de nombreuses approches existent pour formaliser une structure ludique *(p.68)*. Chaque usine à jeux propose donc la sienne, qui est généralement unique si on l'observe en détail. Mais dans les grandes lignes, nous avons pu distinguer deux approches principales. Elles sont toutes deux basées sur un principe visant à dissocier la structure ludique en un ensemble « d'objets » représentant les éléments de jeu. Ensuite, nous identifions d'un côté des outils qui proposent des « objets intelligents », qui contiennent à la fois des données et des méthodes, à la manière de l'approche *« Orienté Objet » (p.179)*. De l'autre côté, nous observons des outils qui séparent les données, qui sont contenues à l'intérieur des objets, des méthodes, qui sont centralisées dans un grand script global *(p.178)*. Cette seconde approche rappelle la manière dont sont présentées les règles des jeux de plateau, proche du mode de pensée d'un Game Designer *(p.258)*.

- **Pour le « Compute », éditeur visuel ou langage de programmation ?** Les éditeurs visuels permettent de créer des programmes informatiques sans avoir recours à un langage de programmation. Ils sont donc bénéfiques pour l'accessibilité de l'outil, en particulier auprès d'un public novice *(p.180)*. D'un autre côté, les outils basés sur un langage de programmation sont un excellent moyen d'apprentissage de l'informatique *(p.187)*. L'emploi d'un langage de programmation commun peut même constituer un tremplin vers le monde professionnel de la création vidéoludique *(p.186)*.

Au-delà de ces grandes lignes, les usines à jeux que nous avons étudiées ont chacune leur propre approche de la création vidéoludique. Il s'agit d'outils très différents, aussi bien en

terme d'utilisation que des jeux qu'ils permettent de créer. Loin d'être péjorative, cette diversité permet au contraire d'exposer une large variété de public à la création vidéoludique. Elle permet de s'adapter à différents contextes : âge et niveau d'expertise technique du public, temps dont il dispose…mais aussi la personnalité de chaque créateur. En effet, comme nous le rappelait ***Crawford*** *(p.84)*, la création vidéoludique comporte une part artistique. Chaque créateur utilisera donc des outils différents selon sa propre sensibilité et méthodes de réflexion. Si les propos de ***Crawford*** visaient à expliquer la multitude d'outils théoriques existants, nous pensons qu'ils peuvent également s'appliquer à la diversité des outils techniques que nous avons observée dans ce chapitre.

Cette richesse d'approches de facilitation de la création vidéoludique inhérente aux outils techniques ne bénéficie pas qu'au champ du divertissement. En effet, nous avons pu observer plusieurs exemples d'utilisation d'usines à jeux à des fins utilitaires. D'un côté, certaines expérimentations utilisent des outils techniques pour faciliter la création de jeux de divertissement par des apprenant. L'activité de création vidéoludique est alors associée à une finalité utilitaire : l'apprentissage d'un contenu pédagogique. Par exemple, ***Céline Dunoyer*** utilise *Quandary* pour écrire des fictions interactives avec des élèves de 6e *(p.166)*, ***Jacob Habgood*** s'appuie sur *Stagecast Creator* pour travailler avec des enfants de 7 ans dans un cadre extrascolaire *(p.180)* et ***Mark Overmars*** enseigne l'informatique à ses étudiants d'université avec *Game Maker* *(p.179)*. Mais il semble également possible d'utiliser les usines à jeux pour créer des Serious Games, telle que l'illustre notre expérimentation avec *The Games Factory 2* *(p.195)*. La diversité inhérente au champ des usines à jeux semble donc permettre l'utilisation de la création vidéoludique à des finalités autre que le divertissement.

Mais les liens entre ces outils techniques de *« Game Design »* et le secteur du Serious Game ne s'arrête pas à ces quelques expérimentations. Nous avons également observé que certains outils de création de Serious Games semblent ouvertement inspirés d'usines à jeux. Ainsi, *SGTools* reprend les bases de *Virtools* *(p.182)*, *e-Adventure* s'inspire librement de *Adventure Game Studio* *(p.174)*, alors que *Thinking Worlds* applique à la 3D temps réel des recettes éprouvées par des outils de création de jeux d'aventure *(p.174)*. Cette influence des outils techniques de Game Design sur les outils de création de Serious Games ne se limite pas au champ des usines à jeux, mais concerne également le secteur du « modding ». Ainsi, *Virtuoso* est un « mod » de création vidéoludique construit sur *Half-Life 2*, alors que *Flip* et *Adventure Author* font de même pour *Neverwinter Nights 2* *(p.172)*.

Au final, la diversité des usines à jeux a déjà inspiré plusieurs applications à finalité utilitaire. Ces initiatives pionnières nous invitent à poursuivre une telle approche, pour éventuellement essayer de l'amener plus loin. Dans la suite de notre travail, nous tenterons donc d'explorer deux directions :

- L'utilisation d'usines à jeux existantes pour la création de Serious Games *(p.248)*.
- Le fait de s'inspirer des usines à jeux pour créer des outils techniques dédiés à la création de Serious Games *(p.214 et p.256)*.

LE « JEU 2.0 »

Au-delà des outils de « modding » et des usines à jeux, notre corpus référence également des logiciels issus d'un courant plus récent : le *« Jeu 2.0 »*[59]. A quoi correspond t'il exactement ?

1 DEFINIR LE « JEU 2.0 »

Parmi les pistes qu'explore actuellement l'industrie du jeu vidéo de divertissement se trouve l'importance du *« player-generated content »* (Sotamaa, 2010). Cet anglicisme fait tout simplement référence au fait que le joueur peut créer du contenu pour le jeu vidéo auquel il est en train de jouer[60]. En effet, depuis quelques années il est possible de voir certaines grosses productions de l'industrie du jeu vidéo proposer des fonctionnalités permettant au joueur de modifier le jeu. Par exemple, *LittleBigPlanet* ***(Media Molecule, 2008)*** et *Halo 3* ***(Bungie, 2007)*** proposent des éditeurs de niveaux, alors que *Spore* ***(Maxis, 2008)*** permet au joueur de dessiner lui-même certains objets du jeu. Alors certes, comme nous l'avons vu dans le chapitre sur le *« modding »* *(p.50)*, ces approches sont loin d'être nouvelles dans leur principe. Elles se différencient pourtant dans leur forme. Tout d'abord, il est maintenant possible de voir des jeux incorporant des outils de modification sur support console, qui plus est sur des titres majeurs. Mais surtout, ces outils s'accompagnent d'un système permettant de partager directement les créations avec d'autres joueurs. Ces fonctionnalités de création et de partage sont clairement mises en avant par les campagnes marketing de ces jeux, comme en témoignent les publicités pour *LittleBigPlanet 2* ***(Media Molecule, 2011)***[61]. Pour appuyer de telles campagnes marketing, le terme de *« Jeu 2.0 »* a été avancé afin de différencier ces jeux qui permettent aux joueurs de « créer et partager » des autres titres (Le Roy, 2006; Nations, 2008). Ce terme est bien évidemment une référence à l'expression *« Web 2.0 »*, qui désigne la vague de sites Internet permettant aux utilisateurs de créer et partager du contenu, par exemple des photos avec *FlickR* ou des vidéos avec *YouTube* (O'Reilly, 2005).

1.1 UNE DEFINITION THEORIQUE DU JEU 2.0

Cependant, ces définitions ne nous renseignent pas véritablement sur la nature du « contenu » qui sera créé par les joueurs dans le cadre du « Jeu 2.0 ». Un chercheur de l'université des arts de Vienne s'est intéressé à ce phénomène. Dans sa thèse, ***Tolino*** analyse un corpus de 160 « créations de joueurs » (Tolino, 2008). Ces créations sont de plusieurs sortes, allant de la construction de nouveaux niveaux pour un jeu donné à la création de films utilisant des séquences capturées d'un jeu *(appelées « machinimas »)*, en passant par la réalisation de déguisements à l'effigie de héros de jeux vidéo *(phénomène du « cosplay »)* ou encore la production de guides de jeu détaillés *(appelés « FAQs »)*. De cette analyse, ***Tolino*** déduit une taxinomie détaillée des différents types de « créations de joueurs » (Tolino, 2009). Pour notre étude, la différence que fait ce chercheur entre les *« artefacts ludiques » (« ludic artifacts »)* et le *« contenu ludique » (« game content »)* nous intéresse particulièrement. Les *« artefacts ludiques »* regroupent toutes les créations de joueur externes au jeu en lui-même, alors que le *« contenu ludique »* désigne les niveaux, les « mods » et toute autre production directement utilisable dans le jeu vidéo.

[59] Traduction littérale de l'expression *« Gaming 2.0 »* que l'on retrouvera dans les textes en langue anglaise.
[60] D'après la définition de http://en.wikipedia.org/wiki/User-generated_content#Player_generated_content
[61] Par exemple le spot publicitaire européen : http://www.youtube.com/watch?v=ts0PTlvMWVo ou encore la publicité européenne diffusée dans la presse http://www.flickr.com/photos/playstationblogeurope/5375447644/

En référence au mouvement du « Web 2.0 », il serait tentant de labelliser comme « Jeu 2.0 » toute création de joueur, qu'elle soit externe ou interne au jeu. Or, toutes les créations de type *« artefact ludique »* sous forme électronique se trouvent généralement appuyées par des sites de la vague du « Web 2.0 ». Puisqu'elle est par définition externe au jeu en lui-même, une *machinima* sera partagée grâce à *YouTube*, pendant qu'une photo de déguisement de *cosplay* se retrouvera sur *FlickR*, alors qu'un guide détaillé viendra enrichir la base de données de *GameFAQs*. Ainsi, la création *« d'artefacts ludiques »* pourrait tout simplement se définir comme l'utilisation du « Web 2.0 » au sein de la culture vidéoludique. Le cas est différent pour les productions de type *« contenu ludique »*, qui s'appuient nécessairement sur un jeu vidéo pour être utilisées, de par leur nature interne à ce dernier. Pourtant, comme nous l'avons déjà évoqué *(p.49)*, les outils de création de contenu pour des jeux vidéo par des amateurs existent depuis de nombreuses années. Avec le développement d'Internet, les joueurs créatifs se sont même structurés en communautés afin d'échanger leurs créations. Là encore, il ne semble pas pertinent de désigner ces approches de création de contenu diffusé par les joueurs au travers de sites Internet externes au jeu comme du « Jeu 2.0 ». En effet, même si les articles journalistiques sur le sujet (Le Roy, 2006; Nations, 2008) ne proposent pas de définitions claires du « Jeu 2.0 », tous les titres qu'ils citent en exemple permettent à la fois de créer du *« contenu ludique »* et de le partager directement à travers le jeu. Nous proposons alors de **définir le « Jeu 2.0 » comme toute application permettant à un utilisateur de créer, d'échanger et de jouer à un « contenu ludique »**.

1.2 DES OUTILS TECHNIQUES CLASSIQUES QUI DEVIENNENT COLLABORATIFS

Au sein de notre corpus, nous identifions 35 outils techniques de Game Design répondant à cette définition du « Jeu 2.0 ». En comparant ces exemples de « Jeu 2.0 » avec les outils techniques présentés précédemment du côté des usines à jeux *(p.166)* ou du modding *(p.50)*, nous constatons que le « Jeu 2.0 » n'apporte pas fondamentalement de nouvelles possibilités créatives *(voir détail du corpus p.202)*.

Par exemple, nous pouvons remarquer que *Addventure*, *Halo 3*, *Super Smash Bros. Brawl*, *LittleBigPlanet*, *Metadragon*, *I Love Your Game*, *Sploder* ou encore les trois outils de ***Cartoon Network*** sont tout simplement des « éditeurs de niveaux ». Dans une logique similaire, *Spore*, *The Sims Carnival Swapper* et *Ugengames* reposent uniquement sur des « éditeurs de graphisme ». Mais nous trouvons aussi des exemples proposant plusieurs éditeurs, comme *The Sims Carnival Game Creator*, *PlayCrafter*, *Wario Ware: Do It Yourself* ou encore *BYOND*. Ces derniers sont donc soit des usines à jeux, soit des ensembles complets d'outils de modding. Il en va de même pour *The Sims Carnival Wizard*, qui permet de créer des jeux en utilisant uniquement des *« menus de choix prédéfinis » (p.160)*. Ce logiciel pose plusieurs questions au créateur en lui offrant à chaque fois une série de réponses limitées. Tout d'abord, le créateur doit choisir entre plusieurs « genres de jeux » disponibles *(par exemple « Course »)*, puis entre différents sous-genres *(par exemple, « Course en vue de dessus »)*. Plusieurs « thèmes visuels » sont ensuite proposés. Une nouvelle série de questions vient offrir la possibilité d'affiner les réglages du jeu *(par exemple choisir entre l'objectif « Gagner la course » ou « Dernier survivant », définir le nombre de tours de circuits à accomplir, régler la puissance des lois physiques…)*. Le créateur peut même modifier l'apparence des principaux éléments de jeu en les choisissant dans une librairie d'images *(par exemple choisir entre plusieurs modèles de voiture de course)*. Enfin, et il s'agit là d'une fonctionnalité des plus étonnantes car très rare dans notre corpus, cet outil ne propose pas *« d'éditeur de contenu émergent »* pour la création de l'état initial mais une nouvelle liste de choix. Le concepteur devra donc choisir un niveau *(par exemple un circuit)* entre plusieurs modèles, sans avoir la possibilité de le construire par lui-même. Si cette philosophie de création permet à l'utilisateur de créer facilement et rapidement un jeu vidéo, ses choix

créatifs s'en trouvent considérablement limités, à l'image du courant des « menus d'options » *(p.53)*. Ce logiciel n'est pas le seul exemple de « Jeu 2.0 » à favoriser cette approche, puisque *Fyrebug* et *GameBrix Express* emploient une logique similaire. Cette philosophie de simplification extrême n'est pourtant pas caractéristique du « Jeu 2.0 », puisque nous la retrouvons dans *The 3D Gamemaker* ***(The Games Creators, 2001)***. Cette usine à jeux proposait même un mode de « création automatique » *(p.184)*.

Nous pourrions alors simplement **définir le « Jeu 2.0 » comme la rencontre entre des outils techniques de création vidéoludique existant depuis plusieurs années et des plateformes de partage inspirées du « Web 2.0 »**. Pour autant, les Game Designers amateurs n'ont pas attendu l'arrivée du « Jeu 2.0 » pour partager leurs créations. Comme évoqué précédemment, de nombreux sites amateurs permettent de partager des créations *(cf. exemples de DoomWorld p.50 et de The Daily Click p.178)*. De tels sites de partages sont même parfois le fruit des éditeurs de jeu, en la forme de plateforme officielles *(cf. exemples de Dawn of War II p.51 et de YoYoGames p.179)*. Mais dans tous ces exemples, les plateformes de partage ne sont pas intégrées au logiciel de création. Il faut quitter ce dernier afin d'échanger et/ou de jouer à ses créations. Ainsi, l'apport du « Jeu 2.0 » n'est pas à chercher du côté des capacités de création mais du côté « partage ». Au sein d'un logiciel unique, il est dorénavant possible de créer, échanger et jouer à du *« contenu ludique »*. Cela nous renvoie à l'histoire du « Jeu 2.0 », détaillée en annexe afin de ne pas alourdir ce chapitre *(p.311)*.

Au final, le courant du « Jeu 2.0 » permet donc d'encourager les expériences collaborative de création vidéoludique. En facilitant les échanges entre créateurs, un même projet de jeu peut être conçu par plusieurs personnes différentes. Par exemple, avec *The Sims Carnival*, tout Internaute peut, en cliquant sur un bouton, modifier un jeu existant sur la plateforme de partage et en publier une variante. La plateforme de partage gardant une trace du jeu « originel », il est possible d'observer des créateurs qui ne se connaissent pas oeuvrer collectivement à la réalisation d'un même jeu vidéo. Pour cela, il suffit que chaque créateur travaille à tour de rôle sur la version créé par un des participants au projet. Dans un registre similaire, chaque partie du jeu *Spore* contient des éléments qui auront été créés par différents joueurs. Lorsqu'un joueur lance une nouvelle partie, le jeu va récupérer sur la plateforme d'échange des objets imaginés par d'autres joueurs, tels que des créatures ou des bâtiments, et les utilise pour construire un monde unique. S'il le souhaite, le joueur peut même modifier ces créations provenant d'autres joueurs, avant de les partager à nouveau.

2 ANALYSE D'UN CORPUS D'EXEMPLES DE « JEU 2.0 »

Comme nous l'avons évoqué, en nous appuyant sur cette définition, nous pouvons identifier dans notre corpus un ensemble de **35 exemples de « Jeu 2.0 »**. Afin d'analyser les approches de facilitation de la création vidéoludique inhérente à ce courant, nous présentons le détail de ces 35 outils techniques. Comme les autres outils, ils ont été classifiés avec le *modèle ISICO étendu* *(p.160)*. Deux tableaux distincts sont alors utilisés. Le premier recense les exemples d'outils destinés à la *« modification d'un jeu existant »* alors que le second se focalise sur ceux permettant la *« création de nouveaux jeux autonomes »*. Rappelons que lors de la constitution de notre corpus, l'accent a clairement été mis sur les « usines à jeux » au détriment des outils de « modding » *(p.162)*. Le premier tableau ne recense donc que quelques exemples illustratifs de « Jeu 2.0 » liés au *« modding »*, alors que le second présente un panorama plus large « d'usines à jeux » réinventées selon les principes du « Jeu 2.0 ». Chaque tableau référence les différents critères enregistrés pour les quatre « parties » de notre modèle classificatoire, assorties d'informations quantitatives sur le nombre de jeux *(ou de « contenu ludique »)* produits et partagés par les utilisateurs de chacun de ces outils.

Tableau 12. Détails des exemples de « Jeu 2.0 » de la catégorie « modification de jeux existants »

Outil	Genre	Initial State	Input	Compute	Output	Partage
Addventure ***(Allen S. Firstenberg, 1994)***	Aventure & Rôle	Editeur visuel	-	-	Editeur visuel Représentation textuelle	86.372 épisodes[62] créés avant fermeture aux contributeurs
Halo 3 ***(Bungie, 2007)***	Tir & Action	Editeur visuel	-	Configuration de paramètres	- Représentation 3D	20.000 niveaux et 20.000 variantes de jeu[63].
LittleBigPlanet ***(Media Molecule, 2008)***	Plateforme	Editeur visuel	-	-	Importation de fichiers Représentation 3D	3.915.131 niveaux[64]
Metadragon ***(Metamorphes, 2009)***	Autre genre	Editeur visuel	-	Configuration de paramètres	Sélection dans librairie Représentation 2D	89 niveaux[65]
Spore ***(Maxis, 2008)***	Autre genre	-	-	-	Editeur visuel Représentation 3D	160.540.309 objets[66]
Super Smash Bros. Brawl. ***(Nintendo, 2008)***	Combat	Editeur visuel	-	-	-	Fermé le 06-30-09. Plateforme officieuse avec 14.794 niveaux[67]
Wario Ware : Do It Yourself ***(Nintendo, 2009)***	Tous genres	Editeur visuel	Divers (pad)	Editeur visuel	Editeur visuel Représentation 2D	Partage pair-à-pair entre consoles ou via *WiiWare*

Tableau 13. Détails des exemples de « Jeu 2.0 » de la catégorie « création de nouveaux jeux autonomes »

Outil	Genre	Initial State	Input	Compute	Output	Partage
Batman : The Brave and The Bold Game Creator ***(Cartoon Network, 2009)***	Plateforme	Editeur visuel	-	Configuration de paramètres	- Représentation 2D	Plus de 6 millions de jeux[68].
Ben 10 : Alien Force Game Creator ***(Cartoon Network, 2008)***	Plateforme	Editeur visuel	-	Configuration de paramètres	- Représentation 2D	
Star Wars : The Clone Wars Game Creator ***(Cartoon Network, 2009)***	Tir & Action	Editeur visuel	-	Configuration de paramètres	- Représentation 2D	
BYOND 4.0 ***(Dantom, 2007)***	Tous genres	Importation de fichiers	Clavier Souris	Langage de script propriétaire	Importation de fichiers Représentation 2D	608 jeux[69]

[62] Relevé le 07-03-11 à partir de http://www.addventure.com/addventure/

[63] Relevé le 05-03-11 à partir de http://www.bungie.net/online/. Cependant la plateforme de partage ne conserve que le 20.000 derniers fichiers envoyés : les fichiers les plus anciens datent de seulement trois mois. Le jeu étant sorti en 2007, si nous estimons la production des joueurs à 20.000 niveaux et variantes de jeux chaque trois mois, nous pouvons estimer qu'environ 2.400.000 niveaux et 2.400.000 variantes de jeux ont été crées au total.

[64] Relevé le 05-03-11 à partir de http://lbp.me/search?t=newest

[65] Relevé le 05-03-11 à partir de http://www.metaphormes.com/

[66] Relevé le 07-03-11 à partir de http://www.spore.com/sporepedia

[67] Relevé le 07-03-11 à partir de http://supersmashbros.ign.com/

[68] Chacun des trois outils du groupe *Cartoon Network Game Creator* ne conserve que les 200 derniers jeux créés. Mais, d'après un communiqué de presse datant du 12-10-09, 6 millions de jeux ont été créés et partagés pour ces trois outils. Relevé le 05-12-09 à partir de http://news.turner.com/article_display.cfm?article_id=4712

[69] Relevé le 04-03-11 à partir de http://www.byond.com/games/

Challenge You ***(ChallengeYou, 2007)***	Autre genre	Editeur visuel	-	-	- Représentation 3D	38.895 jeux[70]
DUNG ***(Dantom, 1996)***	Aventure & Rôle	Importation de fichiers	Clavier Souris	Langage de script propriétaire	Importation de fichiers Représentation 2D	Remplacé par *BYOND*
Fyrebug ***(Fyrebug, 2007)***	Tous genres	Sélection dans librairie	-	Configuration de paramètres	Importation de fichiers Sélection dans librairie Représentation 2D	Fermé[71]
GameBrix Builder ***(DigitalBrix, 2007)***	Tous genres	Editeur visuel	Clavier Souris	Editeur visuel	Importation de fichiers Représentation 2D	Fermé[72]
GameBrix Express ***(DigitalBrix, 2008)***	Tous genres	Sélection dans librairie	-	Configuration de paramètres	Importation de fichiers Sélection dans librairie Représentation 2D	Fermé[73]
GameSalad ***(Gendai Games, 2008)***	Tous genres	Editeur visuel	Clavier Souris	Editeur visuel	Importation de fichiers Représentation 2D	3978 jeux[74]
Gamestar Mechanic ***(GameLab, 2009)***	Tir & Action	Editeur visuel	-	Configuration de paramètres	Sélection dans librairie Représentation 2D	10.330 jeux[75]
Game Studio ***(IJsfontein Interactive Media, 2007)***	Plateforme	Editeur visuel	-	Configuration de paramètres	Sélection dans librairie Représentation 2D	21.007 jeux[76]
I Love Your Game ***(OUAT Entertainment, 2010)***	Autre genre	Importation de fichiers	-	-	Importation de fichiers Représentation 2D	102 jeux[77]
Kodu Game Lab ***(Microsoft, 2009)***	Tous genres	Editeur visuel	Divers (pad)	Editeur visuel	Sélection dans librairie Représentation 3D	Partage P2P sur *Xbox 360*. 421 jeux[78] sur site *PC* officieux
MyGameBuilder ***(Jolly Good Ideas, 2007)***	Aventure & Rôle	Editeur visuel	-	Configuration de paramètres	Importation de fichiers Editeur visuel Représentation 2D	9.455 jeux[79]
Picapoc ***(Picapoc, 2011)***	Gestion Stratégie	Editeur visuel	-	Editeur visuel	Importation de fichiers Sélection dans librairie Représentation 2D	38 jeux[80]
Playcrafter ***(ZipZap Play, 2008)***	Tous genres	Editeur visuel	-	Configuration de paramètres Langage de script propriétaire	Importation de fichiers Sélection dans librairie Représentation 2D	56.002 jeux[81]
Popfly Game Creator ***(Microsoft, 2008)***	Tous genres	Editeur visuel	Clavier Souris	Editeur visuel Langage de programmation commun	Importation de fichiers Sélection dans librairie Représentation 2D	Fermé le 24-08-2009[82].
Scratch ***(MIT Media Lab, 2007)***	Tous genres	Editeur visuel	Clavier Souris	Editeur visuel	Importation de fichiers Editeur visuel Représentation 2D	1.612.511 projets[83]

[70] Relevé le 04-03-11 à partir de http://www.challengeyou.com/PlayGames.php
[71] Relevé le 04-03-11 à partir de http://www.fyrebug.com/
[72] Relevé le 04-03-11 à partir de http://www.gamebrix.com/
[73] Relevé le 04-03-11 à partir de http://www.gamebrix.com/
[74] Relevé le 04-03-11 à partir de http://gamesalad.com/games/
[75] Relevé le 04-03-11 à partir de http://gamestarmechanic.com/game/alley
[76] Relevé le 05-03-11 à partir de http://gamestudio.hetklokhuis.nl/gamepodium/allegames
[77] Relevé le 05-03-11 à partir de http://www.iloveyourgame.com/fr/top/most-recent
[78] Relevé le 05-03-11 à partir de http://planetkodu.com/category/games/
[79] Relevé le 05-03-11 à partir de http://s3.amazonaws.com/apphost/MGB.html
[80] Relevé le 05-03-11 à partir de http://www.picapoc.com/page/liste-des-jeux
[81] Relevé le 05-12-09 à partir de http://www.playcrafter.com/
[82] Relevé le 05-03-11 à partir de http://popflyteam.spaces.live.com/blog/cns!51018025071FD37F!336.entry
[83] Relevé le 25-02-11 à partir de http://stats.scratch.mit.edu/community/

Sharendipity ***(5D Research, 2009)***	Tous genres	Editeur visuel	Clavier Souris	Editeur visuel	Importation de fichiers Sélection dans librairie Représentation 2D	1.127 jeux[84]
Sploder ***(Geoff Gaudreault, 2007)***	Plateforme Tir & Action	Editeur visuel	-	-	- Représentation 2D	1.726.341 jeux[85]
StoryLand ***(Alessandro Schillaci, 2005)***	Aventure & Rôle	Editeur visuel	-	-	Editeur visuel Représentation textuelle	Fermé[86]
StorySprawl ***(Curt Siffert, 1998)***	Aventure & Rôle	Editeur visuel	-	-	Editeur visuel Représentation textuelle	Fermé[87]
The Sims Carnival Wizard ***(Electronic Arts, 2008)***	Tous genres	Sélection dans libraire	-	Configuration de paramètres	Sélection dans librairie Représentation 2D	Fermé le 16-01-11. Hébergeait 15.294 jeux[88] créés avec un de ces trois outils.
The Sims Carnival Swapper ***(Electronic Arts, 2008)***	Tous genres	-	-	-	Importation de fichiers Représentation 2D	
The Sims Carnival Game Creator ***(Electronic Arts, 2008)***	Tous genres	Editeur visuel	Clavier Souris	Editeur visuel	Importation de fichiers Représentation 2D	
Ugengames ***(MobiTween, 2007)***	Tous genres	-	-	-	Importation de fichiers Représentation 2D	3.316 jeux[89] avant fermeture.
WhoseGame ***(Orange, 2007)***	Plateforme	Editeur visuel	-	Configuration de paramètres	Importation de fichiers Sélection dans librairie Représentation 2D	Fermé le 28-02-11. Hébergeait 1.125 jeux[90]

Nous avons déjà évoqué la grande similarité entre les outils du « Jeu 2.0 » et les usines à jeux et outils de modding antérieurs à ce courant *(p.201)*. En revanche, une des spécificités du « Jeu 2.0 » est sa dimension collaborative, qui permet d'ailleurs d'évaluer facilement la popularité des outils de notre corpus. Puisque tous ces outils intègrent une plateforme de partage, pour mesurer le succès rencontré par un outil donné, il suffit de compter le nombre de créations qu'elle propose.

Si nous remarquons que plusieurs logiciels sont « fermés », nous constatons surtout de très gros écarts entre les différents outils. Par exemple, là où *Spore* héberge plus de 160 millions de créations de ses joueurs, *Metadragon* n'en rassemble que 89. Pourtant ces deux outils, bien que différents, sont tous deux très faciles à utiliser. L'éditeur de *Spore* permet au joueur de créer des objets en 3D pour divers éléments du jeu *(les créatures, les bâtiments, les véhicules…)*. Quiconque a déjà pu essayer un logiciel professionnel de modélisation 3D sait à quel point ce genre d'outil peut être très complexe à utiliser. Afin de rendre l'acte créatif accessible et amusant pour le plus grand nombre, les concepteurs de *Spore* ont choisi une approche de création différente. Au lieu de proposer à l'utilisateur de « sculpter » bout par bout un modèle 3D comme dans les outils professionnels, il va tout simplement assembler des « morceaux » de manière à créer un modèle. Cette logique de « puzzle » permet d'inventer de nombreuses créations *(il s'agit donc d'un « éditeur de contenu émergeant »)*, tout en restant utilisable sans expérience préalable en modélisation 3D. Cet outil de création se doit également d'être amusant dans son utilisation, *Spore* étant un jeu vidéo commercial du secteur

[84] Relevé le 07-03-11 à partir de http://www.sharendipity.com/browse/
[85] Relevé le 27-02-11 à partir de http://www.sploder.com/
[86] Relevé le 07-03-11 à partir de http://www.terradif.net/online/cyoa/interactive/storyland/en/index.php
[87] Relevé le 07-03-11 à partir de http://www.storysprawl.com/
[88] Relevé le 05-12-09 à partir de http://www.simscarnival.com/
[89] Relevé le 05-12-09 à partir de http://www.ugengames.com/
[90] Relevé le 07-12-09 à partir de http://www.whosegame.com/

du divertissement. Mais la seule qualité de l'outil ne peut expliquer les différences entre le succès de ces deux applications du « Jeu 2.0 ». En effet, l'éditeur de niveau de *Metadragon* est tout aussi agréable et amusant à manipuler. Il permet de créer très rapidement et en quelques clics de nouveaux labyrinthes pour ce jeu. Alors pourquoi si peu de gens l'ont utilisé ? La réponse serait plutôt à chercher dans la dimension commerciale du projet, en particulier sa taille. Sous cet angle, il apparaîtrait presque injuste de comparer ces deux outils. *Metadragon* a été créé par une toute petite entreprise française à titre expérimental. A l'inverse, *Spore* a été le « gros titre » de la rentrée 2008, commercialisé en grande pompes par ***Electronic Arts***, un des plus gros éditeurs de l'industrie du jeu vidéo de divertissement.

Mais dans le domaine du « Jeu 2.0 » le succès n'est pas forcément lié à la taille de l'éditeur du jeu. Ainsi, ***Electronic Arts*** a mis un terme à sa plateforme *The Sims Carnival* après trois ans d'exploitation. Visiblement, le fait que cette plateforme accueille un total de créations aux alentours des 15.000 jeux n'a pas suffit à l'éditeur pour poursuivre l'expérience. Il en va de même pour l'opérateur de téléphonie ***Orange***, qui a fermé *WhoseGame* après quatre ans d'existence et un peu plus de 1000 jeux créés avec l'outil dédié. Il est vrai que si l'on compare ces deux derniers exemples à des projets similaires en taille et en fonctionnalités, la différence de succès est flagrante. Par exemple, pour la même durée d'exploitation que *WhoseGame*, *Sploder* rassemble plus de 1.700.000 jeux. Quant aux trois outils du groupe *Cartoon Network Game Creator*, ils ont accueilli plus de 6 millions de jeux créés par leurs utilisateurs.

Nous voyons là une autre caractéristique qui différencie le « Jeu 2.0 » de ses ancêtres. Qu'il s'agisse d'usine à jeux ou d'outils de modding, une fois que ces outils ont été créés et diffusés leur survie n'est pas directement liée au bon vouloir de son éditeur. En effet, une usine à jeux peut tout à fait continuer à être utilisée même si son éditeur arrête de la commercialiser, ou tout simplement de la diffuser *(cf. exemple de M.U.G.E.N. p.172)*. Par contre, dans le cas du « Jeu 2.0 », l'outil de création est intimement lié à une plateforme de partage. Cette plateforme étant entretenue par l'éditeur du logiciel, si l'éditeur « débranche » la plateforme l'outil de création disparaît avec elle. Faire vivre une plateforme de partage représente un coût financier non négligeable, nous pouvons alors comprendre que pour les « Jeu 2.0 » du secteur du divertissement, la concurrence est très rude. De nombreux outils risquent de disparaître s'ils ne rencontrent pas le succès escompté par son éditeur.

3 APPLICATIONS SERIEUSES DU « JEU 2.0 »

Suite à cette analyse de quelques exemples concrets de « Jeu 2.0 », nous pouvons constater que ce courant est globalement ancré dans la sphère du divertissement. Pour autant, cela n'empêche pas le « Jeu 2.0 » de posséder un potentiel certain pour des finalités utilitaires :

3.1 « JEU 2.0 » ET SERIOUS GAMING

L'approche la plus évidente pour utiliser un outil pensé pour le divertissement à des fins sérieuses semble être celle du « Serious Gaming » *(p.22)*. Après tout, s'il est possible d'utiliser un jeu vidéo pensé pour le divertissement à des fins sérieuses, pourquoi ne serait-il pas possible de faire de même pour des outils de création vidéoludique ?

Comme nous l'avons déjà évoqué, le collectif ***Pedagame*** s'est déjà intéressé à cette question. Ce collectif a par exemple utilisé *LittleBigPlanet* durant une année scolaire complète. Le public visé était constitué de collégiens, et prenait la forme d'ateliers ayant lieu tous les mercredis après-midi. Nous avons demandé à ***Julien Llanas***, qui s'est occupé de ces ateliers, de nous donner ses impressions « de terrain » après avoir détourné *LittleBigPlanet* :

« Pour LittleBigPlanet, nous avons essayé de travailler à la conception d'un niveau avec des élèves qui se sont amusés à construire plusieurs niveaux en autonomie. Mais ce fut assez décevant car aucun n'était vraiment jouable. Cependant, ce genre de travail demande un suivi et une phase de pré-production que nous n'avons jamais menée jusqu'au bout. Les élèves ont fait preuve d'imagination et ont pu tester rapidement certaines de leurs idées, mais aucun résultat abouti n'a été réalisé. »

Cette relative déception est liée à plusieurs difficultés constatées par l'enseignant :

« Les difficultés rencontrées sont diverses :
- Nécessité de rentrer en mode « projet », ce que ne font pas les élèves de collège sans un encadrement contraignant. Cela se traduit par une difficulté à les faire travailler dans le long terme sur un projet de développement de niveau, ainsi qu'à les motiver pour un travail préparatoire car ils veulent manipuler et créer sans passer par une phase de planification, brainstorming et « Game Design ».
- Difficulté à leur enseigner les bases du « Game Design ».
- Difficulté à les faire travailler en équipe.
- Difficulté à les faire travailler sur un projet de création de niveau avec une succession de temps de travail de 55 minutes (temps d'installation et de rangement du matériel inclus).
- Difficulté à les motiver pour un travail de conception qu'ils ne perçoivent pas comme divertissant.

Par ailleurs nous avons également fait un peu de « Level Design » en créant des questions pour le jeu Buzz! Quiz TV grâce à un serveur dédié, et cela a été beaucoup plus facile. »

Au-delà de ces aspects, des questions très pratiques se posent quand on propose un outil de création à un public novice : celui de l'apprentissage de l'outil. Dans le cadre de leur expérimentation avec *LittleBigPlanet*, l'équipe de ***Pédagame*** s'est appuyée sur les cours intégrés au logiciel :

« Pour LittleBigPlanet, les enseignants ont d'abord suivi les tutoriaux, puis ce fut le tour des élèves. Cela a grandement facilité la tâche de tout le monde lors des séances de création de niveaux.
L'aspect génial du jeu est la possibilité de travailler avec différentes matières (plusieurs sortes de tissus par exemple), ainsi que de créer ses propres « stickers » avec des copies d'écran recadrées. La conception de niveaux avait donc un aspect « décoration d'intérieur » qui était assez prenant. Sans compter que tous les objets sont déjà connus du joueur qui a du les récolter dans les niveaux classiques avant de les utiliser dans la création de niveaux. De plus, il n'y a aucune interaction à définir car le principe de jeu reste le même : de la plateforme à la Mario.
Un autre côté très divertissant est la possibilité de jouer avec des interrupteurs, des déclencheurs, des tremplins, des moteurs, des ballons (un peu comme dans The Incredible Machine[91]), ce qui rend rapidement la conception des niveaux très concrète. Au niveau de l'outil lui-même, il n'y a donc pas eu de points bloquants. A

[91] *The Incredible Machine* ***(Dynamix, 1992)*** est un jeu de puzzle dans lequel le joueur doit construire des machineries complexes pour remplir des objectifs simples tel que mettre un ballon dans un panier. Il dispose pour cela de nombreux éléments (ventilateurs, générateur de courant, courroies…) soumis aux lois de la physique et interagissant entre eux (un ventilateur fera monter un ballon). Le jeu propose aussi un mode de construction libre. Ce côté ludique de « jeu de construction » lui aura valu un grand succès et plusieurs suites.

> *la limite, peut-être aurait-il été intéressant de pouvoir éditer des niveaux existants pour les compléter ou les détourner. »*[92]

Si nous synthétisons le retour de cette expérimentation menée auprès de collégiens, nous notons plusieurs points très importants :

- L'apprentissage de la manipulation de l'outil a été facilité par la présence de « cours interactifs » à l'intérieur du jeu.
- De même, la découverte des divers éléments utilisables pour la construction des niveaux a été assurée de manière très ludique : en jouant aux niveaux livrés avec le jeu, le joueur découvre *(et doit même gagner)* tous les blocs qu'il utilisera ensuite pour créer.
- Le fait que l'outil soit agréable à manipuler, que son utilisation soit quelque peu ludique, est un facteur de motivation fort pour des créateurs amateurs.
- Dès que la tâche de création devient ambitieuse et dépasse le simple stade « d'amusement », il faut s'appuyer sur des méthodes de travail en mode « projet » avec des séances de planification en amont de la réalisation.
- Ces phases de planification font appel à un savoir théorique de « Game Design » qui est absent du logiciel.
- Cela rend la tâche de création beaucoup plus longue et nettement plus « laborieuse » que « ludique », ce qui tend à impacter négativement l'attention d'élèves jeunes.
- En revanche, l'utilisation d'outils de créations plus modestes, destinés à la création de contenus plus simples mais plus rapides à réaliser, permet de contrebalancer cet impact négatif. ***Llanas*** nous évoque ici la création de quiz avec le jeu *Buzz! Quiz TV* ***(Relentless Software, 2008)***, qui a été beaucoup plus facile à mettre en place avec ses élèves que la création de niveaux en 3D avec *LittleBigPlanet*.

En écho aux études qui ont été menées sur d'autres outils de modding *(p.163)*, nous pourrions déduire de ces observations que le choix de l'outil importe beaucoup, et doit être mis en adéquation avec le public et les objectifs pédagogiques visés. Nous voyons ici que, dans un atelier qui visait à sensibiliser les élèves au « Game Design » tout en les amenant à travailler en groupe, un outil simple permettant de créer des réalisations modestes s'est avéré plus pertinent qu'un outil plus élaboré permettant d'obtenir des créations plus riches. Dans un autre contexte ou avec un autre public (par exemple plus âgé), le résultat aurait sans doute été différent. **Cela nous renvoie à l'importance d'avoir une large gamme d'outils techniques à disposition, afin de pouvoir proposer des activités de création vidéoludique à des publics divers et variés.**

Nous notons également l'importance de la maîtrise de l'outil par l'enseignant, qui peut tout à fait apprendre à manipuler un outil de la même manière que ses élèves. L'expérience de ***Pédagame*** montre que les tutoriaux intégrés à l'outil sont efficaces, mais qu'ils ne remplacent pas pour autant l'enseignant. De plus, cet exemple particulier illustre le fait que la plupart des outils techniques de notre corpus n'ont pas de véritable lien avec les aspects plus théoriques du « Game Design » *(p.59)*. Cela renvoie à un questionnement plus profond sur les relations entre « outils techniques » et « outils théoriques » de Game Design, sur lequel nous reviendrons dans la dernière partie de cette thèse *(p.263)*.

Dans un registre similaire, mentionnons le projet *Gameplay*, en cours d'expérimentation lors de la rédaction de cette thèse. Encadrées par une équipe d'animateurs de l'association ***Le Cube***, cinq classes d'enfants d'écoles primaires de la région parisienne se voient proposer de réaliser leur propre jeu vidéo pour tablette tactile *Ipad*. Pour cela, ces enfants âgés de 8 à 11 ans s'appuient sur *GameSalad*. Au départ, les enseignants choisissent un thème pédagogique, tels que l'astronomie ou l'écologie, qui sera à la base du jeu de leurs élèves.

[92] Echanges personnels par mail avec ***Julien Llanas***, du 23-12-09 au 28-12-09

Ensuite, les enfants sont guidés dans le processus de création par des animateurs lors de séances en classe. Chaque séance est dédiée à une étape particulière, telle que la définition d'un concept de jeu, la création des graphismes, ou la création des niveaux. Bien que *GameSalad* possède déjà une plateforme de partage, les organisateurs de ce projet pédagogique ont créé un *« blog »* qui permet aux enfants des différentes écoles d'échanger plus facilement *(et en français)* sur l'avancement respectif de leurs jeux vidéo (Balian, 2010). Ce projet pédagogique, qui propose à des enfants de créer eux-mêmes des jeux vidéo à caractère éducatif dans un contexte scolaire, n'est pas sans rappeler l'expérimentation de ***Kafai*** et le projet *KIDSIM* *(p.237)*.

3.2 « JEU 2.0 » ET SERIOUS GAMING « OFFICIEL »

Tous les cas de « Serious Gaming » liés au « Jeu 2.0 » ne sont pas forcément des initiatives personnelles de détournement de la part de pédagogues et autres professionnels extérieurs au monde du jeu vidéo. En effet, conscients du potentiel pédagogique inhérent à tout outil créatif, les éditeurs de « Jeu 2.0 » incitent ouvertement les enseignants à utiliser leurs logiciels pour l'éducation.

Cela peut tout simplement prendre la forme de « kits d'utilisation en classe », tel celui proposé par ***Microsoft*** pour son logiciel *Kodu*[93]. Il s'agit d'un ensemble de « leçons » prêtes à l'emploi que les enseignants peuvent utiliser pour construire des activités pédagogiques avec leurs élèves. La mise à disposition de ce genre de manuels pour l'utilisation de *Kodu* en classe a inspiré le gouvernement Australien à évaluer son efficacité. Dans le cadre d'une vaste étude sur l'apport des technologies du « Web 2.0 » pour l'éducation, *Kodu* a été évalué auprès de 25 classes d'enfants de niveau primaire, collège et lycée *(les élèves interrogés durant l'étude avaient entre 5 et 16 ans)*. Il ressort de cette étude que la plupart des enseignants ayant accepté d'essayer le logiciel ont obtenu des résultats positifs par rapport à leurs attentes (Innovation & Next Practice Division, 2010). Ils ont globalement pu observer que cet outil permettait de mettre en place des activités pédagogiques incitant au travail de groupe et à la réflexion collective. De même, ce logiciel a été unanimement considéré comme pertinent pour développer les capacités de recherche et résolution de problèmes auprès des élèves. En écho à une étude menée sur le logiciel *Alice* *(p.187)*, plusieurs enseignants ont remarqué que des élèves qu'ils considéraient comme « faibles » se sont révélés particulièrement brillants et motivés durant ces ateliers. Certains de ses élèves se sont même vu confier un rôle « d'expert » par l'enseignant. Il leur proposa d'aller aider les groupes rencontrant des difficultés pour créer leur jeu, renforçant de facto la confiance en soi de ces élèves dits « faibles ». Cependant, des limites qui rejoignent les propos de ***Llanas*** ont également été évoquées. En particulier, la contrainte de temps imposée à de tels ateliers en contexte scolaire joue en défaveur de l'activité. Créer un jeu est une tâche qui prend du temps, d'autant plus si l'élève doit découvrir le fonctionnement de l'outil avant de s'y adonner. Ainsi, dans certains établissements où les élèves n'avaient qu'une heure par semaine pour se familiariser avec *Kodu*, de nombreux apprenants ont abandonné en cours de route. Les enseignants responsables de l'activité pensent que ces étudiants n'ont tous simplement pas eu le temps d'appréhender le logiciel, et donc de pouvoir réellement l'utiliser pour créer ce qu'ils voulaient. Dans une telle situation, le choix d'un logiciel plus simple, mais nécessitant un apprentissage moins long car possédant moins de fonctionnalités, aurait sans doute été judicieux.

Enfin, au-delà des manuels d'utilisation pédagogique, certains éditeurs de « Jeu 2.0 » participent à l'organisation de concours de création visant à proposer des utilisations « éducatives » de leurs outils. Financés par des entités telles la ***MacArthur Foundation*** et le

[93] Disponible en anglais à l'adresse http://fuse.microsoft.com/project/kodu.aspx

réseau éducatif ***HASTAC***, la *Digital Media and Learning Competition* est un concours organisé chaque année pour récompenser l'usage pédagogique des nouvelles technologies. Dans l'édition 2010 de ce concours est apparue une catégorie *« Game Changers »*, avec une stipulation très simple : utiliser soit *LittleBigPlanet* soit *Spore: Galactic Adventures* ***(Maxis, 2008)***[94] pour créer un niveau « éducatif ». Cette catégorie propose une dizaine de prix tels que *« meilleur niveau pour élèves de collège »* ou *« meilleur niveau pour l'apprentissage de la physique »*. Des dotations allant de 5.000$ à 50.000$ ont été remises aux neuf projets qui ont remportés un prix à ce concours[95]. Par exemple, dans le projet *A Day in the Life of a Computer* ***(Gemixin, 2010)***, les joueurs découvrent le fonctionnement interne d'un ordinateur avec des puzzles permettant d'aborder la notion de codage binaire ou la programmation événementielle. Et tout cela à travers un niveau pour le jeu *LittleBigPlanet*. En sponsorisant ce concours, ***Sony*** et ***Electronic Arts***, respectivement éditeurs de *LittleBigPlanet* et de *Spore*, ont donc clairement incité au détournement de leurs logiciels pour des usages « sérieux ».

3.3 LE SERIOUS GAME COMME AVENIR DU « JEU 2.0 » ?

Au-delà de ces pratiques s'apparentant au « Serious Gaming », la majorité des « Jeu 2.0 » de notre corpus se destinent pourtant au secteur du divertissement. Or, comme nous l'avons évoqué précédemment, ce secteur très concurrentiel ne semble pas bénéfique à tous les « Jeu 2.0 ». En effet, dans les 31 exemples « récents » (sortis entre 2007 et 2011) de notre corpus, 10 applications ont déjà fermé. En quatre ans, cela représente un tiers d'outils qui ont fermé faute de rencontrer un succès suffisant. Ces outils malchanceux ne sont pas tous le fruit de petites entreprises, car des géants de l'industrie comme ***Electronic Arts***, ***Microsoft***, et ***Orange*** ont également fermés leurs « Jeu 2.0 ». Les considérations d'ordre budgétaires ne semblent donc pas être un facteur suffisant pour expliquer l'échec de ces outils. Nous ne pensons pas non plus que cela soit directement lié à leur qualité. En effet, les 10 outils ayant fermés, en particulier ceux proposés par des grandes entreprises, sont vraiment puissants et relativement simples d'accès, en tout point comparables aux usines à jeux dont ils s'inspirent. 7 de ces outils étant gratuits et distribués par Internet, le facteur prix ne semble pas non plus être déterminant. Nous pourrions également penser qu'il s'agit d'un manque d'intérêt général pour le « Jeu 2.0 », mais quand l'on constate que certains outils dépassent allègrement le million de créations de joueurs, cet argument semble lui aussi limité. Alors, pourquoi autant d'outils du « Jeu 2.0 » ont-ils fermés leurs portes en si peu de temps ? Comment expliquer que des outils de création vidéoludique apparemment proches connaissent des destins si différents ?

S'il semble difficile de comparer un à un les 31 éléments récents de notre corpus, nous proposons d'illustrer cette situation par la comparaison de deux outils : le *Cartoon Network Game Creator*, qui regroupe trois outils ayant engendrés plus de 6 millions de jeux, et *The Sims Carnival*, qui regroupe trois outils ayant engendré 15.294 jeux sur la même période, et qui est dorénavant fermé par son éditeur faute de succès suffisant. Ces deux outils sont sortis en 2008. Ces deux outils sont gratuits. Ces deux outils s'appuient sur la technologie *Flash*. Ces deux outils bénéficient de « licences fortes » d'un point de vue marketing, la série des *Sims* d'un côté, et trois dessins animés à succès de l'autre. Clairement, ces deux outils sont en tous points comparables à une exception près : leurs possibilités créatives. En effet, du côté de *Cartoon Network Game Creator* ces dernières sont plus que limitées. Chaque outil ne permet de créer des jeux que pour un genre donné. Mais surtout, l'acte de création se résume à la construction d'un niveau qui ne peut occuper qu'un seul écran, éventuellement assorti du réglage de quelques paramètres pour les règles de jeu. Du côté de *The Sims Carnival*, si deux des outils proposés sont aussi simples que limités, le troisième est une véritable usine à jeux

[94] *Spore: Galactic Adventures* est une extension du jeu *Spore* qui permet de créer des « aventures » en la forme de missions scénarisées, qui sont partageables de la même manière que les objets créés avec le jeu de base.
[95] Les différents gagnants sont visibles ici : http://dmlcompetition.net/year_3/winners.php?comp=gc

très complète. *The Sims Carnival Game Creator* permet de créer tous les genres de jeux, et possède de véritables *« éditeurs de contenu émergeant »* pour chacune des composantes du *modèle ISICO*. Il est donc possible d'utiliser cet outil pour créer des jeux longs et complexes, voire même pour inventer de nouveaux jeux.

A la lecture de ce comparatif, nous pourrions légitiment nous demander pourquoi la situation n'est pas inversée. En effet, c'est ici l'outil le plus puissant qui est le moins utilisé. Cette situation peut se comprendre en se replaçant dans le contexte du divertissement. Créer un jeu riche et complexe est une tâche longue et laborieuse, même si l'outil choisi est simple à utiliser. A l'inverse, créer un jeu composé d'un simple niveau en posant quelques blocs sur une grille est très rapide. Si créer un jeu intéressant avec *The Sims Carnival* demandera au minimum quelques heures de travail à un joueur, en faire de même avec *Cartoon Network Game Creator* ne durera vraisemblablement pas plus de quinze ou vingt minutes. Et qu'importe si l'apport créatif est très limité. La grande majorité des joueurs trouvent visiblement plus amusant de passer quelques minutes à « faire son jeu », même si ce dernier ressemble beaucoup à celui des autres joueurs, que de passer des heures et des heures à créer quelque chose d'unique. De plus, un outil qui propose de nombreuses options créatives sera plus long à appréhender qu'un outil limité à un seul des aspects du *modèle ISICO*. Dans le secteur du divertissement, quand un logiciel est distribué gratuitement sur Internet, une des principales sources de financement est la publicité. Et pour obtenir de bonnes recettes publicitaires, il faut une large audience. Nous comprendrons alors qu'un éditeur de jeu vidéo de divertissement préfère posséder un « Jeu 2.0 » à la puissance créative limitée mais qui possède une audience de 6 millions de créateurs, plutôt qu'un outil très puissant qui attire « seulement » 15.294 créatifs. La comparaison de ces deux logiciels illustre bien les différences d'audience entre les divers exemples de « Jeu 2.0 » de notre corpus. Que ce soit *Spore* et ses 160 millions d'objets, *LittleBigPlanet* et ses presque 4 millions de niveaux, *Sploder* et ses plus d'un million et demi de jeux ou même l'ancêtre *Addventure* et ses plus de 80.000 « épisodes », tous ces outils créatifs sont volontairement limités à un seul des aspects du *modèle ISICO*. Même s'ils permettent de créer de nouveaux jeux, ils ne permettent pas d'en contrôler tous les aspects. A l'inverse, des outils puissants permettant de créer tous les aspects d'un jeu avec des *« éditeurs de contenu émergeant »* récoltent une audience très faible, à l'image de *GameSalad* et ses 4000 jeux ou de *Sharendipity* et son millier de jeux, quand ils ne ferment pas carrément leurs portes comme *Popfly* et *GameBrix Builder*.

A la lumière de ces éléments, **le secteur du divertissement ne semble donc pas avoir de la place pour les outils du « Jeu 2.0 » les plus puissants**. S'agit-il pour autant d'une situation immuable ? Un élément de notre corpus nous permet d'entrevoir un autre voie : celle du « Serious Game ». En effet, *Gamestar Mechanic* est un « Jeu 2.0 » qui n'est pas destiné au divertissement. Cet outil a été réalisé par des chercheurs pour l'école primaire *Quest to Learn* et son cursus centré sur la dimension ludique. En complément du développement de méthodes pédagogiques basées sur le jeu, un des objectifs de cette école expérimentale est d'enseigner la culture ludique. Ainsi, l'objectif pédagogique affiché par *Gamestar Mechanic* est d'aider à sensibiliser les élèves au « Game Design ». Cet outil se présente tout d'abord sous la forme d'un jeu d'aventure en ligne, dans lequel chaque utilisateur incarne un avatar. A travers ses explorations, le joueur débloque des « composants » qu'il peut ensuite utiliser pour créer ses propres jeux. Un atelier de création de jeu vidéo permet à chaque joueur de réaliser les titres vidéoludiques qu'il imagine, et de les partager avec les autres membres du site. Pensé pour l'usage en classe, *Gamestar Mechanic* permet aux élèves d'exprimer leur créativité tout en découvrant le processus de fabrication d'un jeu vidéo. Un guide d'utilisation pédagogique conséquent accompagne ce titre et donne de nombreuses pistes aux enseignants souhaitant l'utiliser (Salen, 2009). A noter que ce projet est piloté par ***Katie Salen*** et ***Eric Zimmerman***, chercheurs par ailleurs reconnus pour leur travaux fondamentaux sur le « jeu » (Salen & Zimmerman, 2003, 2005). Si *Gamestar Mechanic* n'est pas forcément le « Jeu 2.0 » le plus

puissant de notre corpus, il est en revanche le seul qui semble pouvoir se satisfaire d'avoir rassemblé « seulement » 10.330 créations. En effet, cette application n'est pas destinée à rassembler un maximum d'audience publicitaire, mais tout simplement à permettre à des élèves d'exprimer leur créativité tout en leur apprenant les bases du « Game Design ».

Cet exemple semble montrer que **le « Jeu 2.0 » peut tout à fait s'inscrire en dehors de la sphère du divertissement**. Et pour pouvoir créer des Serious Games, il semble plus que pertinent d'avoir des outils permettant de créer ou modifier toutes les parties d'un jeu définies dans le *modèle ISICO*. Dans un contexte pédagogique, l'amusement de l'utilisateur n'est plus le seul facteur d'évaluation. Un outil de la famille du « Jeu 2.0 » plus long à utiliser mais permettant de créer des jeux plus riches y sera donc a priori le bienvenu, à l'inverse de ce que nous pouvons observer dans le domaine du divertissement. Et si ***Microsoft***, ***Orange*** ou ***Electronic Arts*** ne souhaitent pas relancer leur outils en visant le domaine des « Serious Games », d'autres initiatives semblent prêtent à s'y atteler. Ainsi, une version « 2.0 » du logiciel *Scratch* est actuellement en cours de développement[96]. Dans sa version actuelle, ce logiciel propose déjà 1.612.511 projets, dont la plupart sont des jeux vidéo ou des animations interactives *(p.189)*. Avec une telle base d'utilisateurs, l'arrivée d'une nouvelle version qui accentue encore plus le caractère « Jeu 2.0 » de cet outil promet de rencontrer un succès conséquent. Si la version actuelle du logiciel doit être téléchargée, et s'appuie sur la technologie *Java* pour permettre la consultation en ligne des créations, la version « 2.0 » sera directement utilisable dans un navigateur grâce à la technologie *Flash*, rejoignant ainsi les choix techniques des autres exemples de « Jeu 2.0 ». Au final, *Gamestar Mechanic* et *Scratch* illustrent donc le potentiel que semble présenter le champ du Serious Game pour des applications de type « Jeu 2.0 ». Ces initiatives semblent d'ailleurs faire écho aux « tendances actuelles » du Serious Game que nous avons évoquées *(p.39)*.

[96] Relevé le 08-03-11 à partir de http://wiki.scratch.mit.edu/wiki/Scratch_2.0

4 SYNTHESE : UTILISATIONS DU « JEU 2.0 » POUR DES FINALITES SERIEUSES

Lors de notre analyse des usines à jeux *(p.198)* et des outils de modding *(p.164)*, nous avions pu mettre en évidence plusieurs approches de facilitation de la création vidéoludique. Mais surtout, nous avions également recensé plusieurs exemples d'utilisation de ces deux types d'outils pour des finalités sérieuses, telles que la pédagogie. Après avoir analysé le courant du « Jeu 2.0 » à travers des exemples représentatifs issus de notre corpus *(p.202)*, nous constatons que les outils techniques du « Jeu 2.0 » reprennent globalement toutes les approches de facilitation observées pour les « usines à jeux » et les « outils de modding ». S'ils reprennent les approches de leurs ancêtres, ils innovent en les associant à des plateformes de partage intégrées, favorisant ainsi la dimension collaborative de la création vidéoludique *(p.201)*.

Cette innovation semble avoir favorisé l'utilisation du « Jeu 2.0 » à des fins utilitaires, car nous recensons plusieurs exemples dans ce domaine *(p.206)*. Par exemple, *Scratch* *(p.189)* et *Gamestar Mechanic* *(p.210)* ont été explicitement conçus pour être utilisés en tant que support à des activités pédagogiques. De leur côté, *LittleBigPlanet* *(p.206)* et *Kodu* sont utilisés avec succès par des enseignants pour cette même finalité, bien qu'ils soient originellement destinés au secteur du divertissement. Mais les outils du « Jeu 2.0 » peuvent également être utilisé pour la création de Serious Games. Ainsi *LittleBigPlanet* et *Spore* sont utilisés pour organiser des concours de création de Serious Games *(p.209)*, tandis que *GameSalad* permet à des enfants en école primaire de créer leurs propres jeux à vocation utilitaire avec le soutien de leurs instituteurs *(p.206)*.

Au final, que ce soit pour les outils de « modding » *(p.164)*, les usines à jeux *(p.198)*, ou le « Jeu 2.0 », nous constatons que tous les outils techniques de Game Design et leurs approches de facilitation semblent pertinents pour la création de Serious Games. Pourtant, il existe d'autres outils techniques qui sont explicitement présentés comme permettant la création de Serious Games. Quelles sont alors les différences entres ces outils de *« Serious Game Design »* et tous les outils de création vidéoludique que nous avons analysés jusqu'ici ?

OUTILS DE SERIOUS GAME DESIGN

En plus des nombreux outils techniques de *« Game Design »* de notre corpus, il existe également quelques outils techniques qui sont explicitement destinés au *« Serious Game Design »*. Etant donné que les outils techniques du *« Game Design »* semblent tout à fait pertinents pour la création de Serious Games *(p.164, p.198 et p.213)*, en quoi ces nouveaux outils se différencient-ils de ceux que nous venons d'étudier ?

En écho aux tendances actuelles du secteur du Serious Game *(p.39)*, nous nous sommes focalisés sur les outils destinés à deux secteurs d'application. Nous étudierons tout d'abord des outils techniques destinés à l'industrie du Serious Game, avant de nous intéresser à ceux conçus pour le secteur de l'éducation.

1 APPLICATIONS INDUSTRIELLES

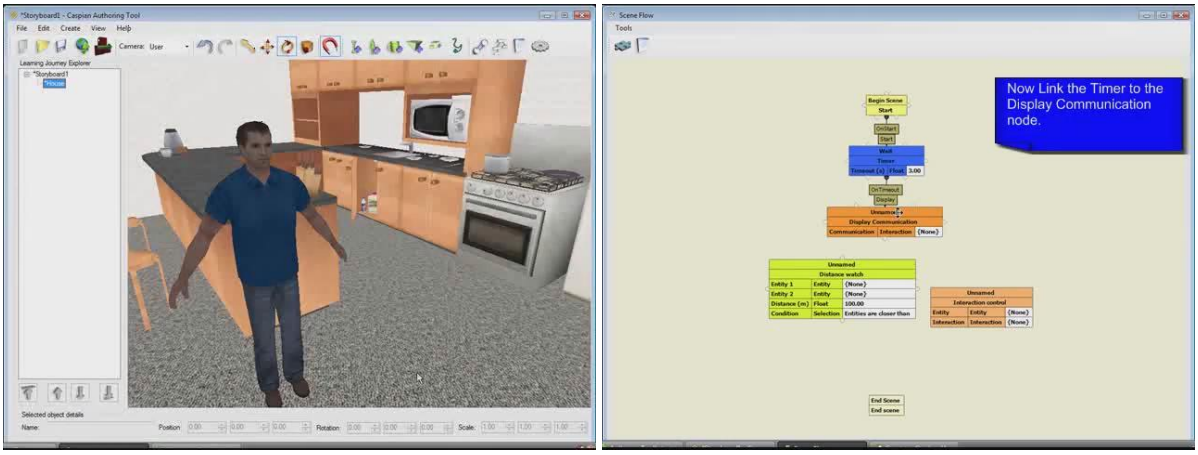
Comme nous l'avons déjà évoqué, les outils techniques destinés à l'industrie du Serious Game visent principalement à accélérer et simplifier le processus de création d'un jeu vidéo afin de réaliser des économies d'échelle *(p.39)*.

1.1 DES OUTILS TECHNIQUES POUR LES PROFESSIONNELS

A cette fin, les outils proposés ont parfois recours à des concepts provenant du Game Design, à l'image de *SGTools* ***(Onlineformapro, 2010)*** qui s'appuie sur le fonctionnement de *Virtools* *(p.182)*. Sans forcément reprendre le fonctionnement d'un outil existant, ils peuvent tout simplement s'en inspirer pour proposer un mode de création plus original. Ainsi, le logiciel *Thinking Worlds* ***(Caspian Learning, 2007-2010)*** emprunte des méthodes identifiables au sein des usines à jeux spécialisées dans les aventures graphiques *(p.174)*, et les développe de manière à proposer un outil qui soit adapté à la création rapide de Serious Games.

Pour créer un Serious Game avec cet outil, l'utilisateur choisi d'abord un environnement 3D, qui aura été préalablement modélisé par des infographistes. Il peut ensuite y ajouter des objets et personnages, là aussi sous formes de modèles 3D qui auront été créés au préalable. Une fois le niveau de jeu ainsi construit, ce logiciel permet de créer l'interactivité du jeu de manière très simple. S'il propose un langage de script propriétaire, la création des règles de jeu est plus rapide et aisée en utilisant l'éditeur visuel dédié. Cet éditeur propose une liste prédéfinie de « blocs », que l'utilisateur peut connecter de manière graphique. Chaque bloc possède une fonction précise, telle que *« déplacer la caméra », « si le joueur est à moins de X mètres de l'objet »*… En assemblant ces « blocs » les uns avec les autres, il est possible de créer visuellement l'interactivité du jeu. Au-delà des déplacements dans un univers 3D, le jeu permet également de créer simplement des systèmes de dialogues interactifs, de quiz, et des petits puzzles qui permettent d'agrémenter l'aventure d'énigmes ou d'informations « sérieuses ». Ce principe de fonctionnement est relativement proche des certaines usines à jeux que nous avons déjà étudiées, notamment de *Virtools* *(p.182)*. *Thinking Worlds* propose par contre une fonctionnalité inédite à tous les outils que nous avons étudiés jusqu'ici : l'intégration des jeux créés à des plateformes *LMS* respectant les normes *SCORM*. Ainsi, il est possible de transformer les jeux en *« activités pédagogiques »*, et surtout de créer un système de suivi du joueur générant un « rapport d'analyse » de sa performance et du transfert de

connaissance entre le Serious Game et l'apprenant *(p.145)*. C'est par cette fonctionnalité que *Thinking Worlds* se distingue des outils techniques de *« Game Design »* de notre corpus.



79. *Thinking Worlds* : éditeur de niveaux *(gauche)* et de règles *(droite)*

Fin 2010, la technologie de *Thinking Worlds* a été couplée à celle du Serious Game *Virtual Battlespace 2* ***(Bohemia Interactive, 2009)***, bien connu du monde de la simulation militaire. Il en résulte le logiciel *VBS Worlds* ***(Bohemia Interactive & Caspian Learning, 2010)***, qui reprend globalement l'approche de création de *Thinking Worlds* mais l'applique au contenu militaire de *Virtual Battlespace 2*. *VBS Worlds* ajoute donc l'approche *« sélection dans une bibliothèque »* de l'interface sortante à *Thinking Worlds* pour accélérer d'autant plus la création de Serious Games destiné à l'entraînement militaire. A noter d'ailleurs que cette spécialisation industrielle de *Thinking Worlds* n'était pas la volonté originelle de son éditeur ***Caspian Learning***. En effet, les premières versions de ce logiciel étaient distribuées gratuitement aux enseignants afin de leur permettre de créer rapidement des Serious Games pour leurs élèves. Si la société britannique s'est par la suite recentrée sur le seul secteur industriel, *Thinking Worlds* semble donc illustrer qu'un même outil peut bénéficier à l'ensemble du secteur du Serious Game.

1.2 RETOUR D'EXPERIENCE : AMELIORATION D'UN OUTIL TECHNIQUE

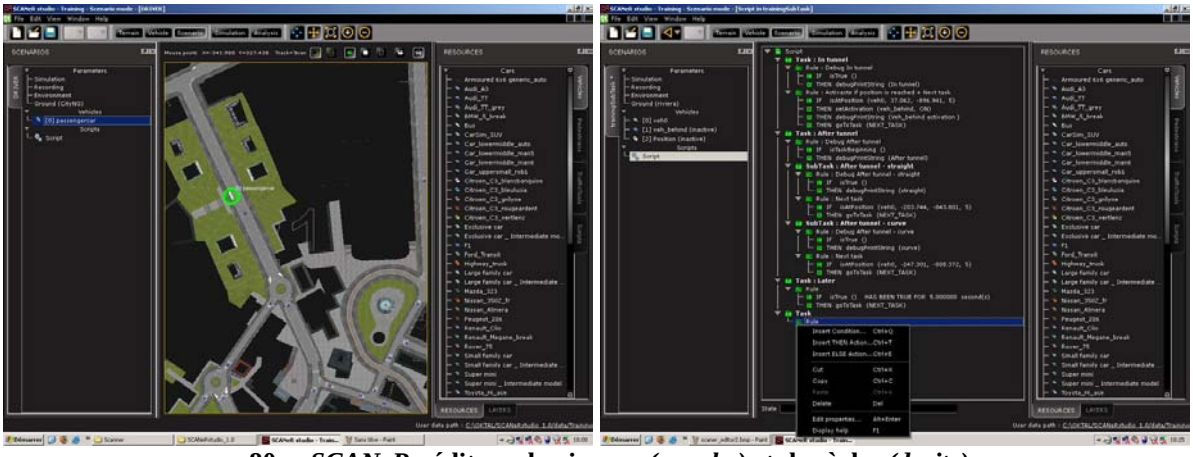
Toutes les sociétés spécialisées dans la création de Serious Game ne commercialisent pas forcément leurs outils de travail. Ainsi, l'outil *SCANeR* ***(OKTAL, 2010)*** est un logiciel utilisé en interne par la société ***OKTAL*** pour créer des simulateurs permettant de prodiguer un entraînement à la conduite de tout type de véhicule routier. Les Serious Games et simulations réalisées avec cet outil couvrent des domaines d'applications très larges, de la formation à la conduite de poids lourds à l'utilisation de véhicules d'entretien en passant par la sensibilisation à l'éco-conduite, à l'image du projet *geDriver* *(p.86)*. Dans le cadre du *CIFRE* accompagnant cette thèse, en plus de la conception d'un Serious Game reposant sur *SCANeR*, nous avons été amené à participer à une réflexion sur les évolutions de cet outil.

1.2.1 PRESENTATION DE SCANER

Issue de l'ingénierie automobile, *SCANeR* est un logiciel modulaire destiné à créer des simulations automobiles pour deux usages : l'ingénierie de véhicules et l'entraînement à leur conduite. La dimension « ingénierie » se retrouve dans le haut niveau de fidélité de la simulation physique du comportement des différents véhicules : *adhérence au sol, courbe de puissance du moteur…* La simulation vise le plus haut niveau de réalisme possible. Cette fidélité dans la simulation du comportement physique des véhicules fait de *SCANeR* un outil de choix pour l'entraînement à la conduite desdits véhicule. Les Serious Games créés avec cet outil sont d'ailleurs compatibles avec de nombreux périphériques dédiés à la simulation, du

simple volant à retour de force branché sur un ordinateur standard à l'habitacle de véhicule monté sur vérins et couplé à un cluster d'ordinateurs et plusieurs vidéoprojecteurs.

D'un point de vue création, *SCANeR* propose un ensemble d'éditeurs permettant de créer des simulateurs d'entraînement automobile, et représente donc une sorte « d'usine à Serious Games ». Au départ, cet outil a été conçu et imaginé pour faciliter le travail des employés d'***OKTAL***, et donc pour réaliser des économies d'échelle sur la création de Serious Games. Cependant, un des objectifs affichés par les ingénieurs travaillant à l'évolution constante de cet outil est de rendre le processus de création accessible à un public novice. Ainsi, les clients d'***OKTAL*** seront en mesure de créer eux-mêmes leurs scénarios pédagogiques à partir de la base définie lors de la création d'un Serious Game avec cet outil. Cette approche « tout public » de *SCANeR* est à rapprocher des outils de « modding » livrés avec un jeu commercial *(p.50)*. Pour un utilisateur non spécialisé, l'outil ne permet pas de créer de nouveaux Serious Games autonomes. Mais il facilite la création de nouveaux scénarios et mises en scène, autrement dit de nouveaux « niveaux », pour le Serious Game qui aura été réalisé par les ingénieurs d'***OKTAL***. Par exemple, la création de nouveaux éléments graphiques ou de véhicules est un processus complexe demandant la fois des compétences en modélisation 3D, en programmation ainsi que des données captées sur un véhicule réel pour le modèle de simulation physique. Par contre, le logiciel propose des outils simples pour la création de nouvelles routes. Il permet ensuite de peupler ces routes par les divers véhicules disponibles dans la bibliothèque d'objets livrées avec le Serious Game. Puis, il est possible de créer des règles de jeu par un éditeur dédié. Cet éditeur se base sur le paradigme des « conditions » et « actions » à piocher dans une liste globale « d'instructions », un peu à l'image du pionnier *Gamemaker* *(p.177)*, mais en plus moderne. Par exemple, il est possible d'associer une condition *« quand le véhicule du joueur arrive à une distance X du véhicule B »* à l'action *« lancer le mouvement du véhicule B »* afin de créer une voiture qui avancera vers le joueur dès qu'il s'en rapproche. Il est ainsi possible de créer des simulations très riches, l'outil permettant de contrôler jusqu'aux feux de circulation. En accord avec la finalité « sérieuse » des jeux créés avec *SCANeR*, de nombreuses fonctionnalités permettent de suivre le joueur pour évaluer sa performance et son apprentissage. De nombreuses variables telles que le régime moteur, la consommation d'essence, la durée d'appui sur une pédale ou même le rejet de *CO2* sont simulées par le jeu. Le concepteur du jeu peut s'y appuyer dessus pour analyser la manière de conduire de l'apprenant, et générer un rapport adéquat qui sera utile au formateur utilisant le Serious Game.



80. *SCANeR* : éditeur de niveaux *(gauche)* et de règles *(droite)*

1.2.2 LES USINES A JEUX POUR AMELIORER UN OUTIL DEDIE AU SERIOUS GAME

Ironiquement, ce travail de simplification du logiciel pour permettre aux clients de modifier les jeux livrés a également suscité un grand intérêt auprès des concepteurs travaillant dans la société. Pour ces derniers, ces avancées représentent un gain de temps considérable dans leur

travail. En lien avec ***Viviane Pena***, conceptrice de simulations pédagogiques au sein de la société ***OKTAL***, nous avons donc été amené à réfléchir aux évolutions à apporter à *SCANeR*. L'objectif de cette réflexion est de rendre l'outil plus accessible, et donc de faciliter son utilisation pour la création de Serious Games. Si ***Viviane Pena*** apportait sa grande expérience dans la domaine de l'e-learning, de notre côté nous avons bien évidement été inspirés par les nombreuses usines à jeux étudiés dans cette thèse. De cette réflexion commune est née trois axes de propositions :

1.2.2.1 Le renforcement de la spécialisation du logiciel

En regard de notre typologie d'outils techniques *(p.159)*, *SCANeR* est un outil spécialisé dans la création d'un « genre » précis de Serious Games : les simulations automobiles. Un des premiers axes de réflexion sur les évolutions de l'outil concerne donc l'étude des fonctionnalités présentes dans ce genre de jeux, dans l'optique d'automatiser leur création par le logiciel. Si de nombreuses fonctionnalités sont déjà présentes, nous avons pu en identifier une qui était absente.

De nombreux clients d'***OKTAL*** souhaitent utiliser leur Serious Games pour former leur personnel à la conduite de véhicules, mais aussi au repérage de lieux précis. Par exemple, lors de la conduite de véhicules d'entretien sur un site précis *(zone industrielle…)*, une bonne connaissance des lieux et de leurs zones dangereuses est primordiale. Nous avons donc recherché des exemples d'applications qui favorisaient le repérage de lieux par des apprenants. Nous avons identifié un exemple inattendu dans l'univers des jeux vidéo de divertissement : *Grand Theft Auto III* ***(DMA Design, 2001)*** et ses suites. En effet, un joueur des épisodes en 3D de la série *GTA* est rapidement amené à mémoriser la géographie de la ville qui lui sert de terrain de jeu. Une des principales activités de ce jeu consiste à se rendre d'un lieu à l'autre dans une gigantesque ville reconstituée. A force de jouer, le joueur finit par connaître cette ville virtuelle par cœur, et devient capable de se rendre à un lieu précis d'où qu'il soit, et ce sans avoir recours à un plan ou autre *GPS*.

81. *GTA San Andreas* : boussole dans l'écran de jeu *(gauche)* et carte complète *(droite)*

Si cet apprentissage est nécessaire à la réussite dans le jeu, nous remarquons qu'il est renforcé par la manière dont le jeu est conçu. En effet, pendant le jeu, un petit *GPS* est constamment affiché en bas à gauche de l'écran. Cependant il est très incomplet et ne sert finalement que de boussole. Pour se repérer dans la ville, le joueur doit mettre le jeu en pause, et afficher une carte complète où est seulement indiquée « sa position » *(croix blanche)* ainsi que ses divers objectifs à atteindre *(les autres icônes)*. En début de partie le joueur se repère donc en affichant régulièrement la carte. Mais avec le temps, il finit par mémoriser la géographie des lieux et n'éprouve plus le besoin de l'afficher. Nous pouvons supposer que si le jeu proposait un système de guidage plus efficace à l'intérieur du jeu, comme une ligne à suivre durant la conduite ou un *GPS* plus grand, la mémorisation des lieux serait moins pertinente pour le

joueur car il lui suffirait de suivre les indications à l'écran *(p.113)*. Alors que le système qui consiste à mettre le jeu en pause pour afficher un plan demande au joueur un certain effort cognitif, ne serait-ce que pour mettre en relation sa position sur le plan 2D avec l'affichage du monde en 3D *(p.114)*. Cet exemple illustre les propos des nombreux chercheurs soulignant le fait que, de par leur nature, tous les jeux vidéo demandent aux joueurs d'apprendre afin de gagner *(p.111)*. Si le fait de connaître la topographie d'une ville virtuelle n'est pas véritablement un savoir des plus « utilitaire », la manière dont *GTA* met le joueur dans une situation l'incitant à apprendre la géographie d'une ville semble tout à fait applicable à des lieux réels. Ainsi, nous avons proposé d'intégrer ces fonctionnalités de mise en pause du jeu pour consulter un plan et de mini *GPS* escamotable dans les Serious Games réalisables avec *SCANeR*. Cela permettra d'utiliser des méthodes pédagogiques éprouvées par des jeux vidéo du divertissement pour transmettre un contenu « sérieux ».

1.2.2.2 L'ajout de fonctionnalités de création d'éléments d'e-learning

Nous observons également que les Serious Games réalisés avec *SCANeR* sont régulièrement utilisés en situation de formation professionnelle. Par exemple, de nombreuses sociétés de formation des conducteurs de poids-lourds utilisent les jeux réalisés par ***OKTAL*** auprès de leurs stagiaires. Des demandes spécifiques à ce contexte d'utilisation reviennent donc régulièrement, et peuvent également représenter une source de fonctionnalités à intégrer dans un tel outil.

Par exemple, les formateurs utilisent régulièrement des formations magistrales couplées à des questionnaires papier pour transmettre des éléments théoriques telles que le code de la route ou des procédures particulières. Lors des sessions d'utilisation de Serious Games avec les stagiaires, ils éprouvent régulièrement le souhait d'évaluer ces connaissances théoriques, en général par le biais de quiz intégrés à la simulation. Si un scénario met le stagiaire dans une situation où son camion tombe en panne, plusieurs choix sur l'attitude à adopter seront proposés : *mettre les warnings, utiliser la radio pour appeler un dépanneur…* L'ajout de fonctionnalités permettant de créer facilement et rapidement de tels quiz, voire la possibilité pour un formateur de créer ses propres quiz pour personnaliser un Serious Game, semblent donc des ajout pertinents à un logiciel tel que *SCANeR*. Nous avons alors imaginé une interface externe pour générer des quiz, ainsi que des « règles » pour les intégrer facilement à la simulation, par le biais du système de « groupes d'instructions » détaillé dans la section suivante.

Un autre type de fonctionnalité demandée est bien évidement la possibilité pour le formateur d'évaluer les performances de ses stagiaires. *SCANeR* permet déjà d'enregistrer les très nombreuses variables qu'il simule, et de les utiliser pour construire un système de rapport d'évaluation. Mais tous les clients n'ont pas forcément besoin de rapports riches, et le fait d'obtenir uniquement quelques variables de base, telles que le temps passé sur l'exercice, le nombre de sorties de route ou la vitesse moyenne, seraient tout à fait suffisante car le formateur est tout à fait en mesure d'interpréter ces variables. Nous avons alors proposé la mise en place d'un système de « débriefing automatique », qui génère un rapport d'évaluation a partir de quelques variables de base, et qui est automatiquement enrichi selon les fonctionnalités utilisées dans le Serious Game. Par exemple, si le système de « repérage par plan » évoqué précédemment est intégré au jeu, alors ce rapport automatique affichera le nombre d'utilisation de la carte par le stagiaire. Cela permettra au formateur d'évaluer la maîtrise des lieux simulés par le jeu, en partant du principe qu'un stagiaire qui se repère sans utiliser la carte a déjà commencé à bien assimiler la topographie des lieux. Pour accentuer le gain de temps de création apporté par cette fonctionnalité, elle est intiment liée à un autre système destiné à accélérer la réalisation d'éléments récurrents dans les Serious Games réalisés avec *SCANeR*, tel que détaillé ci-après.

1.2.2.3 La facilitation du processus de création des règles

Les concepteurs de Serious Games travaillant pour la société ***OKTAL*** ont observé que lors de la réalisation de simulations sur *SCANeR*, certaines tâches revenaient régulièrement, en particulier lors de la création de règles par l'éditeur idoine. Cet éditeur propose un grand nombre « d'instructions » pour construire des règles, allant des conditions permettant d'évaluer la distance entre deux véhicules à la création et modification de variables globales à la simulation. La grande majorité de ces « instructions » sont de relativement bas niveau. Lors de la création d'un Serious Game, le concepteur est donc généralement amené à associer plusieurs de ces instructions pour créer un seul évènement scénaristique. Par exemple, pour réaliser une mise en situation simple telle qu'un véhicule percutant le joueur si celui-ci reste arrêté moins de 3 secondes à un stop, le concepteur devra effectuer les opérations suivantes :

- *Tout d'abord placer un « véhicule percuteur » dans le niveau, définir son mouvement graphiquement grâce à l'éditeur de niveau, et le rendre invisible au départ.*
- *Et ensuite utiliser l'éditeur de règles pour créer les règles suivantes :*
    - *« début du scénario » => « création d'une variable T pour compter le temps, initialisée à la valeur 3 » + « création d'une variable booléenne pour savoir si le joueur est déjà rentré dans la zone »*
    - *« lorsque que le véhicule joueur entre dans une zone (+définition de la zone devant le stop) » => « décrémenter la variable T de 1 toutes les secondes » + « définir la variable booléenne à 'vrai' »*
    - *« si la variable booléenne est égale à 'vrai' » + « si la variable T est supérieure à zéro » + « si le véhicule du joueur ne se trouve pas dans une zone (+définition de la zone devant le stop) » => « faire apparaître le véhicule percuteur devant le joueur » + « lancer son mouvement »*

Si la création de ce comportement est loin d'être complexe grâce au système de saisie « sans programmation » proposé par le logiciel, le fait d'avoir à répéter la démarche de création de toutes ces instructions plusieurs fois par scénario s'avère assez fastidieuse. Nous avons alors étudié les usines à jeux de notre corpus pour voir comment elles répondent à ce type de problématique. Nous avons principalement été inspiré par le fonctionnement des logiciels *The Games Factory 2* et *Virtools*. Concrètement, à la lecture du « pseudo-code » présenté ici, nous pourrions imaginer synthétiser cette suite d'instruction dans un seul groupe d'instruction. Pour être plus efficace, ce groupe d'instruction pourrait recevoir des paramètres globaux, en l'occurrence les coordonnées de la zone de « stop », la durée du décompte et l'identité du « véhicule percuteur » *(le véhicule joueur étant par convention définit comme le « véhicule #0 » dans SCANeR)*. Ainsi, pour créer ce type d'évènements, le concepteur n'aurait plus qu'à effecteur trois opérations :

- *Tout d'abord placer un « véhicule percuteur » dans le niveau, définir son mouvement graphiquement grâce à l'éditeur de niveau, et le rendre invisible au départ.*
- *Appliquer le groupe d'instructions « démarrer mouvement si le joueur reste moins de X secondes dans une zone » sur ce véhicule.*
- *Régler les autres paramètres de ce groupe d'instructions : mettre le décompte à la valeur « 3 » et définir la zone d'arrêt à tester.*

Le gain de productivité peut paraître négligeable sur cet exemple illustratif isolé, mais il est pourtant réel si un processus de réflexion similaire est appliqué à tous les « groupes d'instructions » créés de manière récurrente par les concepteurs. De plus, ces récurrences étant généralement transversales à plusieurs projets, c'est par le biais de ce type de simplification qu'un outil technique permet à des créateurs industriels de Serious Games de réaliser des économies d'échelle. Deux approches ont donc été proposées pour l'évolution de *SCANeR*. ***Viviane Pena*** a tout d'abord identifié les « groupes d'instructions » qu'elle était amenée à créer de manière récurrente dans ses projets. La première approche consiste à demander aux développeurs de l'outil de créer des « comportements » permettant

d'automatiser la création de ces « groupes d'instructions » identifiés par la conceptrice. Une telle approche de facilitation est par exemple mise en place dans *The Games Factory 2*, à travers son système de *« mouvements prédéfinis »* *(p.178)*. La seconde approche serait de permettre directement aux concepteurs de créer eux-mêmes des « groupes d'instructions » réutilisables, posant ainsi les bases d'un système évolutif. Cette approche est quant à elle observable dans le système de *« blocs de comportement »* de *Virtools* *(p.182)*.

1.2.2.4 Synthèse

Au final, ce retour d'expérience du *CIFRE* illustre comment les outils techniques de Game Design peuvent servir de source d'inspiration pour améliorer les outils techniques de création de Serious Games. Les jeux vidéo en eux-mêmes peuvent également être une source d'inspiration comme nous venons également de le voir dans ce retour d'expérience. A noter à ce propos que, contrairement à ***Caspian Learning*** et ***Bohemia Interactive***, ***OKTAL*** n'est pas une société issue de l'industrie du jeu vidéo mais du monde de la simulation industrielle. Pour autant, en comparant *SCANeR* et *Thinking Worlds*, il est impossible de ne pas voir de nombreux points communs entre ces deux outils de création de Serious Games, par exemple dans leur manière de faciliter la création de scénarios et de règles de jeux à travers un « éditeur visuel » ou encore le fait qu'il intègrent des fonctionnalités de suivi du joueur. Ces deux outils possèdent également des différences, comme le fait qu'ils ne soient pas spécialisés dans la réalisation du même genre vidéoludique. Ces différences illustrent quelque part la diversité du champ des Serious Games *(p.18)*, qui rassemble des acteurs d'origines très variées, et qui engendre des jeux permettant aussi bien de diffuser des messages, à l'image de ceux créés avec *Thinking Worlds*, que des jeux visant à prodiguer un entraînement, comme ceux basés sur *SCANeR*.

2 APPLICATIONS PEDAGOGIQUES

Tel que nous l'avons déjà évoqué *(p.41)*, les outils techniques de Serious Game Design destinés à l'éducation cherchent à simplifier la création de jeux vidéo pour la rendre accessible à un public novice en la matière : les enseignants et leurs élèves. Pour cela, ce public doit disposer d'outils permettant de créer de nouveaux jeux ou de modifier des jeux existants sans connaissances préalables en conception vidéoludique. Lors de notre exploration du champ du Game Design, nous avons rencontré de nombreux exemples de tels outils permettant de créer des jeux vidéo en contexte pédagogique. Pour commencer, *Game Maker* *(p.179)*, *Scratch* *(p.189)* et *Gamestar Mechanic* *(p.210)* ont été explicitement conçus pour être utilisés dans le secteur de l'éducation. Mais ce secteur est aussi le siège de nombreux détournements d'usage d'usines à jeux à des fins pédagogiques, par exemple avec *Quandary* *(p.166)*, *Stagecast Creator* *(p.180)*, *LittleBigPlanet* *(p.206)*, *Kodu* *(p.209)*, *Spore* *(p.209)*, et *GameSalad* *(p.206)*. Nous pouvons également retenir les travaux d'***El-Nasr & al.*** sur l'utilisation des outils de « modding » comme support d'apprentissage de l'informatique *(p.163)*.

Au-delà de toutes ces expérimentions sur l'utilisation d'outils de création vidéoludique, des logiciels explicitement destinés aux Serious Games et autres jeux éducatifs commencent à voir le jour. S'ils s'inspirent ouvertement des outils du Game Design, ils apportent également de nouvelles fonctionnalités visant à améliorer leur intégration au sein d'activités pédagogiques, comme l'illustrent les quelques exemples détaillés ci-après.

2.1 VIRTUOSO

A l'origine, *Virtuoso* ***(Oliver Gray, 2007)*** est destiné à permettre la création, par des enseignants, de scénarios pédagogiques pour des jeux d'aventure en 3D. Cet outil est réalisé

par une équipe de chercheurs américains de la ***North Carolina State University***. D'un point de vue technique, la particularité de *Virtuoso* est d'être un « mod » du jeu *Half-Life 2* ***(Valve Software, 2004)***. Ce titre, tournant sur le moteur *Source Engine*, est livré avec un ensemble d'outils permettant de modifier de très nombreux aspects du jeu *(p.51)*. Les chercheurs ont donc utilisé les outils livrés avec ce jeu de divertissement non pas pour créer un autre jeu, mais pour inventer un système de création vidéoludique (Gray & Young, 2007).

Virtuoso propose tout d'abord un « éditeur de niveaux » qui permet de se déplacer librement dans un univers 3D et d'y ajouter des « objets » afin de créer un niveau. Pour palier à la complexité que représente la modélisation d'objets 3D, les enseignants ont à leur disposition une bibliothèque d'objets *(personnages, habitations…)* livrée avec le logiciel. Autre aspect grandement facilité par cet outil, la création du *« Compute »*, et donc des règles de jeu. Elle est basée sur un système de *« condition => action »* relativement proche de celui de *SCANeR* *(p.215)*. Pour chaque objet du niveau, l'utilisateur peut associer des *« comportements »* qui seront composés d'une suite *« d'actions »* à piocher dans un menu déroulant. Ces actions possèdent des paramètres prédéfinis que l'utilisateur peut remplir. Le déclenchement de ces *« actions »* sera conditionné par un *« évènement »*, qui provient également d'un menu déroulant listant toutes les *« conditions de déclenchement »* proposées par l'outil. Par exemple, il est possible d'associer à un objet *« personnage »*, l'action *« avance »* pour la condition *« quand le joueur est à moins de 30 mètres de l'objet »*. L'action *« avance »* propose ensuite un paramètre *« distance »* que l'utilisateur peut régler librement, par exemple pour créer un personnage qui fuit le joueur d'une dizaine de mètres dès qu'il s'en approche. Ce système permet de créer relativement facilement de nombreuses mécaniques de jeu, allant de quiz intégrés à l'aventure à la gestion d'inventaires pour les personnages (Gray, 2008). D'un point de vue technique, les *« actions »* et *« conditions »* sélectionnées par l'utilisateur sont ensuite transcrites dans un script en *LUA* par *Virtuoso*, qui ajoute au *Source Engine* la possibilité d'interpréter ce langage. Au-delà de ces considérations techniques, cet outil illustre parfaitement le fait que les interfaces de programmation reposant sur un « éditeur visuel » peuvent tout à fait être de simples masques appliqués à de véritables langages de scripts. Elles conservent ainsi la puissance d'un langage de programmation malgré la simplification de son utilisation *(p.183)*.

82. *Virtuoso* : éditeur de niveaux *(gauche)* et de règles *(droite)*

De par son statut de projet de recherche, *Virtuoso* intègre également des mécanismes de collecte de données. La progression des élèves qui jouent aux jeux créés par leurs enseignants est enregistrée dans une base de données *mySQL*, un module de communication avec ce système ayant été rajouté au *Source Engine*. Nous retrouvons ici l'importance de la dimension « suivi du joueur » pour le secteur du Serious Game *(p.145)*. Une autre fonctionnalité intéressante de *Virtuoso* est la possibilité de créer des jeux de manière collaborative en temps réel, l'outil bénéficiant des fonctionnalités réseau de *Half-Life 2*. Ainsi, plusieurs utilisateurs

peuvent créer un même jeu en simultané. La seule restriction concerne la création des *« comportements »*, qui ne peut être effectuée que par l'hôte de la session réseau, les autres participants pouvant cependant visionner les *« comportements »* sans les modifier. Lors de l'évaluation de leur outil sur le terrain, les chercheurs ont été surpris de constater que les enseignants se répartissaient spontanément des rôles très précis. Par exemple, l'hôte de la session prenait le rôle du « programmeur » et s'occupait uniquement de créer les comportements, alors qu'un autre enseignant se focalisait sur la création des objets et de leurs propriétés *(taille, couleur…)*. De l'aveu même d'***Oliver Gray***, principal concepteur du logiciel, la plupart des utilisateurs ont préféré la création en équipe selon ce principe de « chaîne d'assemblage en réseau » plutôt qu'en solitaire. Pour autant, les fonctionnalités réseau de cet outil n'ont pas forcément eu qu'un impact positif. Ainsi, le choix de la voie du « modding » compliqua le déploiement de l'outil dans les écoles. En effet, comme tout « mod », *Virtuoso* doit être installé avec le jeu vidéo sur lequel il se base. Or, le système de protection d'*Half-Life 2 (installation de Steam et validation de licence par Internet)* ne fut pas sans poser quelques problèmes techniques sur les réseaux scolaires, sans parler de l'obligation d'acheter autant de licences du jeu que d'utilisateurs de *Virtuoso*. De plus, le mode de création en réseau s'appuie sur le mode de jeu réseau *« Deathmatch »* du jeu originel. Et visiblement, le fait de voir des élèves lancer un logiciel intitulé *« Match à mort »* en classe n'était pas du goût de tous les enseignants, certains d'entre eux remettant alors fortement en question l'intérêt pédagogique de l'outil[97].

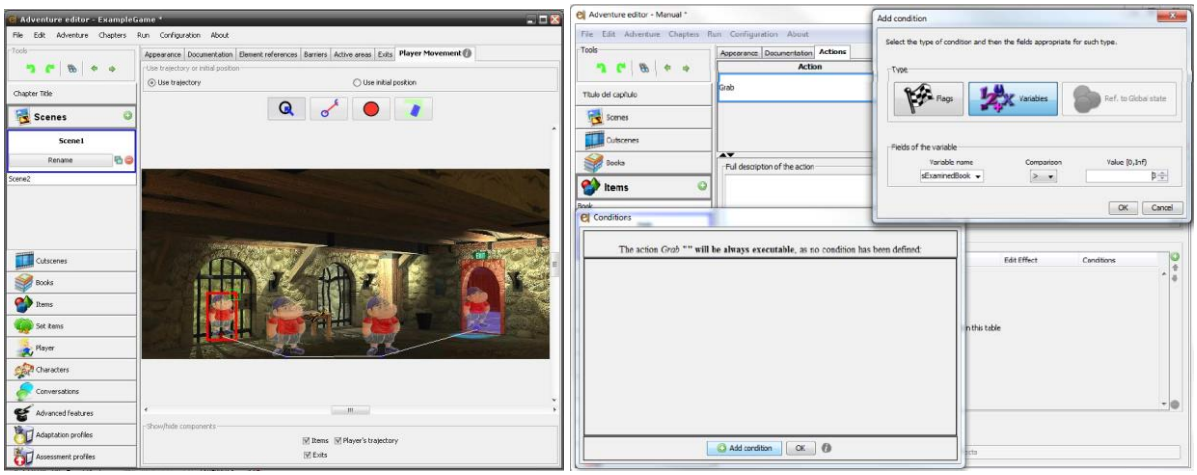
Cet outil a été évalué auprès d'un groupe d'enseignants et de leurs élèves, soit un total d'environ 250 utilisateurs que les chercheurs ont pu observer. Au-delà de la dimension collaborative, une autre de leurs observations est liée à la valeur pédagogique de l'acte de création. Ainsi, le projet prévoyait normalement que seul les enseignants allaient créer des jeux éducatifs, tandis que leurs élèves se contenteraient d'y jouer. ***Annetta & Cheng*** (2008) citent pourtant le cas du jeu *Invicta the Invader*, destiné à des élèves de 6e et traitant des insectes nuisibles que sont les « fourmis de feu ». Au départ, l'enseignant avait réalisé un jeu très riche en contenu, mais aussi très linéaire, que ses élèves trouvaient passablement ennuyeux. L'enseignant proposa alors à ses élèves d'améliorer par eux-mêmes ce jeu. Ce changement de posture amena un gain de motivation considérable de la part des élèves. Afin de modifier le jeu pour le rendre plus amusant, ces derniers ont du assimiler de nombreuses informations sur les « fourmis de feu ». C'est donc l'exercice de création vidéoludique qui a servi de moteur pour l'apprentissage du contenu que l'enseignant souhaitait transmettre, et non la pratique du jeu en lui-même. Ce changement d'approche semble receler un véritable potentiel pédagogique, tel que nous l'étudierons dans le dernier chapitre *(p.235)*. À noter enfin que les qualités de *Virtuoso* lui ont permit de dépasser la sphère du divertissement pour intégrer celle de l'industrie. Le studio ***Virtual Heroes***, renommé pour sa participation à la réalisation d'*America's Army*, a racheté cette technologie et envisage de la commercialiser comme outil d'assistance à la création de Serious Games dans le courant de l'année 2011[98].

2.2 <E-ADVENTURE>

<e-Adventure> ***(Javier Torrente & al., 2009)*** est un outil destiné à faciliter la création de jeux d'aventure éducatifs, réalisé dans le cadre d'un projet de recherche menée à l'université ***Complutense de Madrid***, en Espagne. Les jeux d'aventure créés avec cet outil sont du genre « jeu d'aventure graphique » *(p.174)*. Le joueur y dirige un personnage dans des décors en deux dimensions par le seul intermédiaire de la souris. Il est possible de ramasser divers objets cachés dans les décors, et surtout d'avoir de passionnants dialogues avec d'autres personnages à travers un système de choix multiples.

[97] Source : échanges personnels par mail avec ***Oliver Gray*** du 03-03-2010 au 08-03-2010
[98] Source : échange personnel par mail avec ***Leonard Annetta*** le 28-02-2010

<e-Adventure> fonctionne de manière similaire aux outils les plus reconnus dans la création de jeux d'aventure graphique, comme *Adventure Game Studio* *(p.174)*. Les jeux sont constitués d'un ensemble de *« scènes »* liées entre elles. Chaque *« scène »* est construite avec des *« objets »* et des *« personnages »*, dont l'utilisateur définira le nom, l'apparence graphique… dans une interface dédiée. Il est également possible de définir l'interactivité du jeu en rédigeant des *« règles »* pour chaque *« objet »* et *« personnage »*. Ces *« règles »* sont composées de *« conditions »* et *« d'actions »*. L'utilisateur choisit tout d'abord une *« condition »* dans une liste prédéfinie par le logiciel *(par exemple « lorsque le joueur choisi d'utiliser l'objet »)*. Il peut ensuite y associer diverses *« actions »*, telles que : *ajouter l'objet dans l'inventaire du joueur, créer/modifier des variables, changer de « scène » ou encore lancer une « conversation »*. Elément central de ce type de jeu, les *« conversations »* sont créées par un éditeur dédié qui n'est pas sans rappeler les logiciels permettant de créer des aventures textuelles *(p.166)*. Chaque *« conversation »* est constituée d'un ensemble de *« phrases »*, qui peuvent être affectées à chacun des *« personnages »* déjà définit dans le jeu. Ces *« phrases »* peuvent ensuite être liées entre elles de manière graphique, selon des *« règles »* qui seront créées avec l'éditeur de règles du logiciel. Il est également possible de créer des *« phrases »* spéciales qui proposent plusieurs choix de réponses au joueur. Tous ces outils facilitent la création des dialogues interactifs à choix multiples qui caractérisent le genre du jeu d'aventure graphique (Torrente, Marchiori, & del Blanco, 2008).



83. <e-Adventure> : éditeur de niveaux *(gauche)* et de règles *(droite)*

D'un point de vue technique, <e-Adventure> est développé en *Java* afin d'être multiplateforme. Il est composé d'un « moteur d'interprétation » des jeux, qui fonctionne avec des fichiers *XML*, et d'une « interface auteur », qui permet de créer les jeux de manière graphique. Les jeux générés par l'interface auteur sont enregistrés dans des fichiers *XML* qui pourront être interprétés par le moteur de jeu *(p.257)*. Les jeux créés avec cet outil sont donc exportables sous format autonome, mais également sous forme de « ressources pédagogiques » intégrables à des *LMS* respectant les différentes normes *SCORM*. Pour faciliter cette intégration à des *LMS* *(p.145)*, deux systèmes spécifiques ont été intégrés à l'outil : un module « d'adaptation » et un module « d'évaluation » (Burgos et al., 2008). Dans un contexte pédagogique s'appuyant sur un *LMS*, une activité est en général découpée en trois phases. Tout d'abord, un questionnaire préliminaire permet de déterminer le niveau de connaissance préalable de l'apprenant. Ensuite, une activité d'apprentissage lui est proposée, cette activité se concluant par une évaluation. Les résultats de l'évaluation sont alors enregistrés dans le système de suivi pédagogique. Ainsi, les apprenants ont à leur disposition des activités qui s'adaptent à leurs connaissances, et l'enseignant peut suivre l'évolution de ses élèves en consultant les résultats de leurs évaluations. Dans ce contexte, les jeux réalisés avec <e-Adventure> représentent l'activité pédagogique proprement dite. Lors de la création d'un jeu avec cet outil, il est possible de créer des *« règles d'adaptation »* avec l'éditeur de

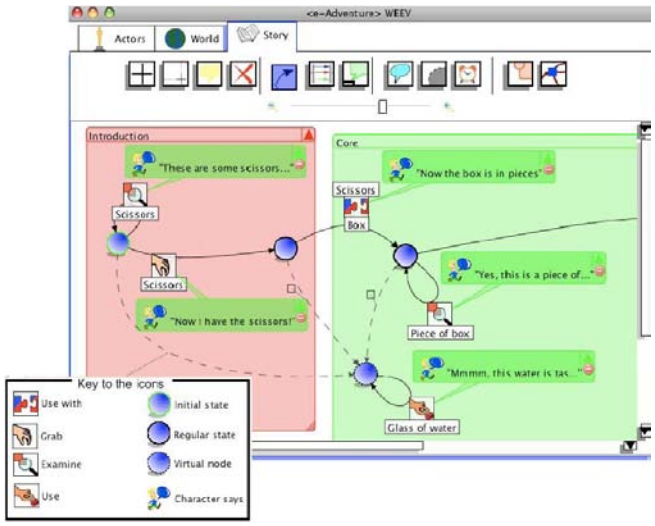
règles du logiciel. Ces règles vont évaluer des paramètres passés au jeu par le *LMS* une fois que l'apprenant aura rempli le questionnaire d'évaluation de ses connaissances préalables. Ces règles permettent alors au créateur de modifier le comportement du jeu selon les réponses des apprenants. Un système similaire est proposé pour la création de *« rapport d'évaluation »*. En utilisation l'éditeur de règles de *<e-Adventure>*, il est possible de définir des critères d'évaluations qui seront communiqués au *LMS*. Ainsi, le concepteur du jeu est libre de définir par lui-même les critères d'évaluation des joueurs *(réponses à des question durant le jeu, temps de jeu, résolution d'une énigme en particulier…)*, ce qui permet de créer des Serious Games selon une approche « intrinsèque » *(p.106)*. Au final, *<e-Adventure>* se distingue donc des usines à jeux spécialisées dans la création des jeux d'aventure graphique par la possibilité d'adapter le jeu au joueur et de définir simplement les critères d'évaluation de sa performance. Loin d'être anecdotiques, ces deux fonctionnalités facilitent grandement l'intégration des jeux créés avec cet outil à un contexte pédagogique.

Parmi les Serious Games réalisés avec *<e-Adventure>* et évalués par les chercheurs, le cas de *HCT Blood Test Game* est particulièrement intéressant (Torrente, Moreno-Ger, Fernández-Manjón, & Blanco, 2009). Ce Serious Game fut développé pour les étudiants en seconde année de médecine à l'université ***Complutense de Madrid***. Il traite du protocole des tests sanguins en laboratoire. Le temps de laboratoire étant rare et les échantillons sanguins étant précieux, l'organisation d'exercices « réels » est très limitée pour les nombreux étudiants en médecine. En partenariat avec les enseignants de l'école de médecine, les chercheurs en informatique à l'origine du projet *<e-Adventure>* ont donc réalisé un Serious Game simulant l'exercice de test d'un échantillon sanguin, et l'ont évalué auprès des étudiants. Un groupe test d'une vingtaine d'étudiants a pratiqué le jeu avant de réaliser l'exercice en laboratoire, alors qu'un groupe de contrôle a réalisé cet exercice sans pratiquer le jeu. Au-delà de l'impact très positif sur la motivation des étudiants, il ressort de cette expérimentation que les étudiants ayant joués au Serious Game avant l'exercice réel maîtrisaient mieux le protocole. De même, le jeu permet de simuler des situations exceptionnelles *(telle que la coagulation d'un tube de sang)*, qui n'apparaissent que très rarement lors de l'exercice en laboratoire, mais auxquelles il faut néanmoins préparer les étudiants. Si les avantages tirés de la pratique de ce jeu sont relativement modestes, il faut les mettre en relation avec l'envergure du jeu. Ici, point de simulation d'un corps humain complet comme dans *Pulse!!* ***(BreakAway, 2007)***. Le jeu proposé est très simple, mais a été déployé sur le *LMS* de l'université de manière à être accessible à tous les étudiants. Sa réalisation n'a mobilisé que deux personnes, en collaboration avec trois enseignants de l'université de médecine qui ont joué le rôle « d'experts » pour le contenu « sérieux » du jeu *(p.150)*. Les chercheurs notent d'ailleurs que ces experts ont pu effectuer des modifications du jeu par eux-mêmes, en utilisant le logiciel *<e-Adventure>*. Cet exemple illustre donc comment la mise à disposition d'un outil permettant à un public non spécialiste de s'adonner à la création de jeu vidéo permet d'impliquer des « experts » dans la création de Serious Game.

Pourtant, comme ils l'expliquent dans un article de synthèse, les chercheurs ont identifiés trois « barrières » à l'introduction de jeux vidéo dans l'éducation (Torrente, Del Blanco, Marchiori, Moreno-Ger, & Andndez-Manjó Andn, 2010). Tout d'abord, la difficulté d'équilibrer les dimensions « ludique » et « sérieuse » *(p.106)*. Pour cela, ils pensent que la mise à disposition d'un outil simple d'accès permet d'amener des spécialistes d'un sujet « sérieux », comme les enseignants, à travailler directement sur le jeu pour s'assurer de ses qualités pédagogiques. La seconde barrière évoquée est liée au coût de création d'un jeu vidéo. Les budgets mobilisés dans l'industrie du jeu vidéo *(p.46)* ou du Serious Game (J. Alvarez, V. Alvarez, Djaouti, & Michaud, 2010) ne sont pas envisageables pour des écoles ou des universités, sauf en cas de financement extérieur. A l'image des tendances actuelles l'industrie du Serious Game *(p.214)*, les chercheurs proposent donc d'utiliser des outils permettant de réaliser des économies d'échelle en mutualisant et automatisant la création de

certaines parties du jeu. Enfin, la dernière barrière évoquée est celle de la diffusion des jeux et de l'évaluation des apprenants. Si grâce aux *LMS* la diffusion est facilitée dans les écoles et universités, la question de l'évaluation reste entière. Dans l'optique d'un jeu basé sur le paradigme « intrinsèque », le contenu pédagogique est indissociable des mécanismes de jeu. La réussite au jeu est donc conditionnée par l'apprentissage du contenu pédagogique. Des critères d'évaluation peuvent donc être intégrés directement à l'intérieur de tels jeux *(p.120 et p.126)*. Comme nous pouvons le voir, les problématiques soulevées par cette équipe de chercheurs semblent trouver une réponse dans les approches de facilitation mises en place par les outils techniques issus du « Game Design » *(p.198)*. L'outil qu'ils ont réalisé, *<e-Adventure>*, reprend donc la grande majorité des fonctionnalités des « usines à jeux », et y ajoute quelques fonctionnalités supplémentaires liées au « suivi du joueur ».

Enfin, après pratiquement deux ans d'utilisation de *<e-Adventure>* pour la réalisation de divers Serious Games, les chercheurs ont observé que les enseignants qui utilisaient leur outil éprouvaient parfois des difficultés à planifier la conception de leur jeu d'aventure (Torrente et al., 2010). Cela nous renvoie à l'utilisation d'outils théoriques pour aider à la conception de Serious Games *(p.150)*. Nous avions déjà observé dans le champ du Game Design que les nombreux outils théoriques disponibles n'étaient que rarement intégrés aux outils techniques. Cette lacune avait déjà été signalée pour l'utilisation d'outils de création de jeu vidéo en contexte scolaire *(p.206)*. Les auteurs de *<e-Adventure>* sont donc en train de réfléchir à l'intégration d'un outil de conception théorique au sein de leur logiciel. Il s'agit d'un outil permettant de modéliser une histoire sous forme de diagrammes. Ces diagrammes permettent d'amorcer la réalisation de certains éléments du jeu, comme les *« personnages »* ou les *« objets »*, tout en guidant le concepteur dans le processus de Game Design. Nous retrouvons ici la logique des outils de modélisation *UML* tels que *Rational Rose* ***(IBM, 2003)*** : en formalisant la conception sous forme de diagrammes, le concepteur prépare l'architecture logicielle de son jeu.

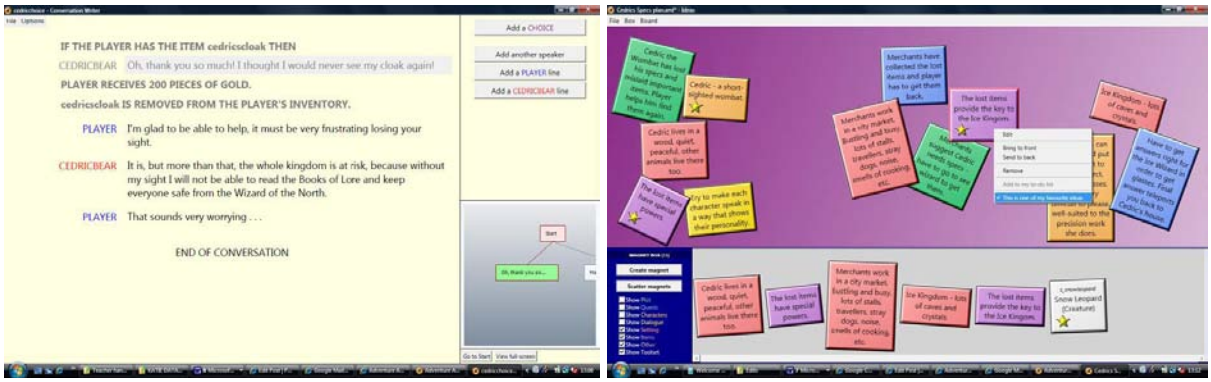


84. *WEEV*, l'outil de modélisation narrative de *<e-Adventure>*

2.3 ADVENTURE AUTHOR ET FLIP

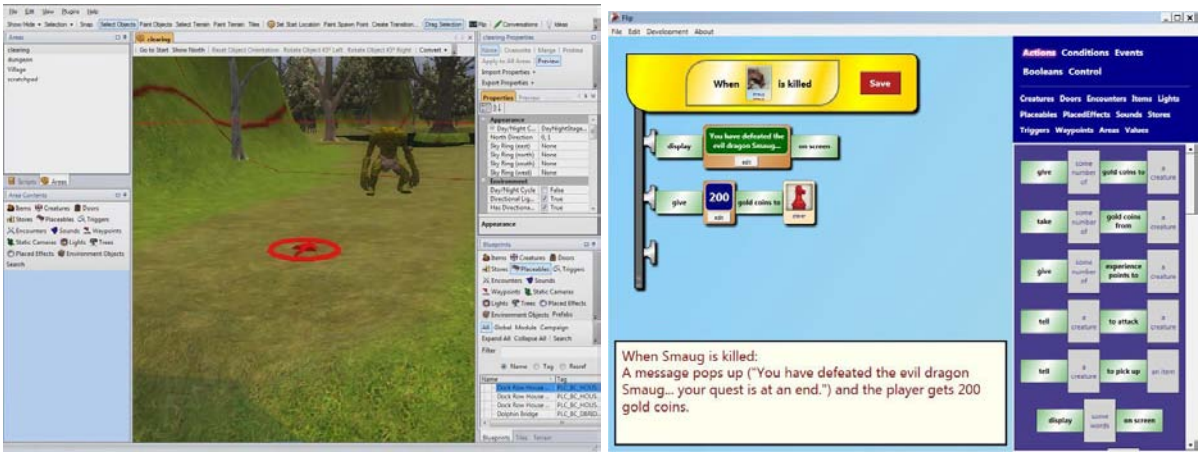
La réflexion sur des approches graphiques permettant de faciliter la conception de jeu d'aventure est également à l'origine de deux autres outils de création de Serious Games pédagogique : *Adventure Author* ***(Judy Robertson & Keiron Nicholson & Cathrin Howells, 2007)*** et *Flip* ***(Judith Good & Keiron Nicholson & Katy Howland, 2010)***. A l'image de *Virtuoso*, ces deux outils sont des « mods » d'un jeu existant, *Neverwinter Nights 2* ***(Bioware, 2006)***. De base, ce titre propose un très puissant « éditeur de niveaux », qui permet de construire des mondes fantastiques en 3D en se basant sur une bibliothèque d'objets pré-établis. Mais l'interactivité, le scénario et les règles du jeu doivent être créés en s'appuyant

sur un langage de script propriétaire. *Adventure Author* propose un « éditeur visuel » pour simplifier cette étape, et la rendre accessible à un public inexpérimenté en matière de programmation informatique (Robertson & Nicholson, 2007). S'inspirant ouvertement des logiciels permettant de créer des jeux d'aventure *(p.166)*, il propose à l'élève d'écrire ses scénarios et dialogues comme une succession de *« phrases »*, liées à des *« choix »* qui amènent vers d'autres *« phrases »*.



85. *Adventure Author* : éditeur de conversation *(gauche)* et « écran à idées » *(droite)*

Ces deux outils open-source sont le fruit de projets de recherche visant à proposer des outils permettant aux élèves de développer leur créativité. La tranche d'âge ciblée en particulier est celle des 10-16 ans. Le souci d'accessibilité de l'outil auprès de ce public a poussé les chercheurs à simplifier les interfaces d'un outil préexistant. Mais les chercheurs s'intéressent également à la démarche de réflexion associée au processus créatif, et à la manière dont un outil logiciel peut l'accompagner *(p.99)*. Ainsi, *Adventure Author* propose un *« écran à idées »* dans lequel les élèves peuvent écrire leurs idées sous forme de post-its. En jouant sur la couleur et la répartition spatiale des post-its, ils peuvent organiser toutes leurs idées, et les mettre en lien avec les différents éléments du jeu qu'ils souhaitent créer.



86. *Flip* : éditeur de niveaux *(gauche)* et de règles *(droite)*

De son côté, *Flip* a bénéficié des expérimentations de terrains menées sur *Adventure Author*. Par exemple, les chercheurs ont constaté que l'automatisation de certaines tâches, bien que facilitant la création de jeu, n'étaient pas forcément pertinentes d'un point de vue pédagogique (Howland, Good, & du Boulay, 2008). Ainsi, *Adventure Author* permet de créer facilement des « personnages » à partir d'une librairie de personnages préexistants. Ces personnages sont générés avec des valeurs par défaut sur leur nombreuses caractéristiques *(nom, taille, habiletés, affiliations…)*. Or, la création de personnages est une phase très importante de l'écriture d'histoires. D'un point de vue pédagogique, le fait que cette phase soit trop simplifiée n'est donc pas forcément bénéfique pour inciter les élèves à travailler les personnages de leur histoire. De même, les chercheurs ont trouvé que les élèves n'étaient pas

assez amenés à rédiger lors de la création de jeu. *Flip* reprend donc la base d'*Adventure Author* et en corrige certaines lacunes. *Flip* conserve l'intégralité des outils de *Neverwinter Nights 2* ainsi que l'éditeur de conversation d'*Adventure Author* et y ajoute un module de création de règles « sans programmation » à destination d'un public novice qui serait rebuté par l'apprentissage du langage de script du jeu. Ce module est plus élaboré que l'éditeur de conversation d'*Adventure Author*, et se rapproche des interfaces graphiques d'initiation à la programmation *(p.187)*. *Flip* s'appuie sur le paradigme « condition / action » rencontré dans de nombreux outils étudiés dans ce mémoire. L'élève associe tout simplement des blocs « SI » et des blocs « ALORS » pour créer l'interactivité de son jeu vidéo (Howland, 2010).

2.4 LES COQUILLES GENERIQUES DE JEUX EDUCATIFS

En dépit de leur simplicité d'utilisation, la création de Serious Games avec les outils présentés jusqu'ici requiert un certain temps de travail. Même simplifiée, la création d'un jeu vidéo demande beaucoup d'éléments, de tests et de corrections. Or, tous les enseignants n'ont pas forcément le temps matériel de réaliser des jeux vidéo aussi élaborés. Face à ce problème, une équipe de chercheurs canadiens a monté un projet permettant aux enseignants de créer très rapidement des Serious Games : les *« coquilles génériques de jeux éducatifs (CGJE) »* (Sauvé & Kaufman, 2010).

L'approche développée ici est de concevoir des modèles de jeux simples mais fonctionnels *(p.100)*, et de proposer à l'enseignant de créer uniquement le « contenu pédagogique » dans un cadre clairement défini. Le processus part de *« coquilles de jeu »* qui sont développées par des programmeurs expérimentés, avec l'assistance de pédagogues pour anticiper au maximum les besoins de « personnalisation » que pourraient ressentir les enseignants. Ces coquilles sont ensuite très facilement modifiables, selon un cadre très précis : l'enseignant ne peut créer ou modifier que le « contenu » du jeu, sous la forme de questions à insérer. L'interactivité et les règles de jeu ne sont pas accessibles, afin de limiter la complexité et la durée du travail de création (Sauvé, 2010b). Par exemple, une des coquilles proposées s'appuie sur le *jeu des petits chevaux*. Les chercheurs ont créé une version vidéoludique de ce jeu de plateau, qui est dorénavant jouable à plusieurs par Internet. Ils ont également légèrement modifié le concept : à chaque fois qu'un joueur désire avancer son pion, il doit répondre à une question. Ces questions sont créées par les enseignants, grâce à un éditeur simple constitué uniquement de champs à remplir. Plusieurs types de questions sont proposées *(questions à choix multiples, questions ouvertes, question vrai/faux…)*, et l'enseignant peut aussi changer quelques éléments graphiques du jeu *(image du plateau…)*. Chaque jeu est ensuite mis en ligne sur la plateforme de partage de l'outil, permettant à chaque enseignant de l'utiliser tel quel dans sa classe, ou de le modifier pour en créer une variante plus adaptée à ses objectifs pédagogiques. A noter également qu'un système d'évaluation des apprenants est intégré au logiciel. Tous les jeux créés avec cet outil enregistrent les bonnes et mauvaises réponses de chaque joueur, afin de permettre à l'enseignant d'évaluer la performance de ses élèves *(p.145)*.

87. *CGJE* : écran de jeu *(gauche)* et question *(droite)*

Entre sa dimension « Jeu 2.0 » pour le partage des réalisations *(p.200)* et la limitation des aspects qui sont créables, cet outil permet de créer un jeu vidéo éducatif en environ deux heures. A ces deux heures de temps moyen de réalisation doivent s'ajouter deux heures supplémentaires de temps préparation du contenu des questions par les enseignants. Au final, l'approche *Coquilles Génériques de Jeux Educatifs* permet à tout enseignant de créer un Serious Game pédagogique en environ quatre heures de travail (Sauvé, 2010c). Comparé au temps requis pour créer un jeu avec les autres outils, cette approche est nettement plus rapide. En contrepartie, elle limite drastiquement la créativité des concepteurs, qui ne peuvent pas toucher aux règles de jeux. Loin d'être meilleur ou plus mauvais que les autres, cet outil est tout simplement différent. Il permet donc de toucher un public d'enseignants qui n'aurait pas eu le temps matériel de s'investir dans des logiciels plus complexes. Ainsi, comme nous l'avons déjà observé *(p.198)*, la diversité des outils existants semble être importante pour faciliter la création de Serious Games auprès de nombreux utilisateurs.

3 SYNTHESE : OUTILS TECHNIQUES DE SERIOUS GAME DESIGN

Au final, les outils techniques destinés au Serious Game sont très proches de ceux étudiés pour les jeux vidéo de divertissement. En guise de conclusion, il nous semble alors pertinent de les analyser eux aussi à travers le *modèle ISICO étendu* *(p.160)*.

Tableau 14. Comparaison des outils techniques de Serious Game Design à travers le *modèle ISICO étendu*

Outil	Type	Genre	Initial State	Input	Compute	Output
Adventure Author	Modification	Aventure & Rôle	Editeur visuel	Clavier Souris	Editeur visuel Langage de script propriétaire	Importation de fichier Sélection dans librairie Représentation 3D
Coquilles Génériques de Jeux Educatifs	Création Partage «2.0»	Autre genre	Editeur visuel	-	-	Importation de fichier Représentation 2D
<e-Adventure>	Création	Aventure & Rôle	Editeur visuel	-	Editeur visuel	Importation de fichier Représentation 2D
Flip	Modification	Aventure & Rôle	Editeur visuel	Clavier Souris	Editeur visuel Langage de script propriétaire	Importation de fichier Sélection dans librairie Représentation 3D
SCANeR	Création	Course	Editeur visuel	Clavier Souris Autres	Editeur visuel	Importation de fichier Sélection dans librairie Représentation 3D
Thinking Worlds	Création	Aventure & Rôle	Editeur visuel	Clavier Souris	Editeur visuel	Importation de fichier Représentation 3D
Virtuoso	Modification	Aventure & Rôle	Editeur visuel	-	Editeur visuel	Sélection dans librairie Représentation 3D

Si l'on compare ce tableau à celui de notre corpus de « Jeu 2.0 » *(p.202)* ainsi qu'aux données statistiques de nos usines à jeux *(p.190)*, nous observons quelques différences d'orientation notables. Là où notre corpus d'usine à jeux est relativement homogène entre usines à jeux généralistes et spécialisées, tous les outils proposés ici sont spécialisés dans un genre donné. Comme pour les usines à jeux spécialisées *(p.192)*, le genre *« Aventure & Rôle »* domine d'ailleurs largement. Plus surprenant, alors que les usines à jeux proposaient une répartition a peu près équilibrée des quatre modes de création du *« Compute »* *(p.193)*, ici l'approche « éditeur visuel » s'impose clairement. Ce fait peut sûrement s'expliquer par la volonté de simplification des outils de manière à rendre accessible la création de Serious Game à un public de novices. Mais contrairement à la sphère du « Jeu 2.0 », cette approche de simplification ne s'est pas faite en limitant le nombre de « parties » d'un jeu créables avec un

outil donné. A une exception près, tous les outils techniques de Serious Game Design que nous avons étudiés permettent de modifier les règles de jeu, alors que dans la sphère du « Jeu 2.0 » des outils aussi puissants ont du mal à rencontrer un large succès public *(p.210)*. Bien que notre corpus d'outils techniques dédiés aux Serious Games soit trop réduit pour pouvoir l'affirmer avec certitude, cette observation semble confirmer ici une tendance observée pour le « Jeu 2.0 » *(p.210)* : **Si les outils de création les plus puissants ont parfois du mal à rencontrer un succès public dans la sphère du divertissement, ils sont pertinents pour le secteur du Serious Game**.

Au-delà de cette observation générale, nous remarquons également que les outils techniques de Serious Game Design proposent souvent une fonctionnalité absente de leurs homologues issus du Game Design : le « suivi des joueurs » *(p.145)*. En effet, si l'analyse des joueurs n'est pas forcément systématique dans les jeux vidéo de divertissement, elle est cruciale dans le champ du Serious Game. A plus forte raison pour les jeux basés sur un paradigme de conception « intrinsèque » *(p.106)*, le Serious Game doit proposer des critères d'évaluation du joueur. Ces critères étant liés aux mécanismes de jeu, il semble donc logique que les outils techniques de Serious Game Design tendent à faciliter l'implémentation de systèmes d'évaluation des joueurs et du transfert de connaissances au sein du jeu.

Pour autant, l'absence d'une telle fonctionnalité n'enlève rien à la pertinence des outils de Game Design pour le champ des Serious Games. D'une part, il reste possible de créer des systèmes d'évaluation de toutes pièces avec les outils de Game Design, même si tâche sera forcément plus complexe qu'avec un outil proposant le suivi des joueurs en standard. Mais surtout, comme le montre notre expérience avec ***Quentin*** *(p.195)*, le fait d'utiliser des usines à jeux généralistes peut être très important quand le genre vidéoludique n'est pas défini. Si l'on y ajoute toutes les expériences d'utilisation d'outils de Game Design à des fins sérieuses que nous avons déjà évoquées *(p.164, p.198 et p.213)*, **nous pouvons alors considérer que l'ensemble des outils techniques de « Game Design » permettent potentiellement de créer des Serious Games**.

BILAN DE LA PARTIE III : CONSIDERATIONS TECHNIQUES SUR LE SERIOUS GAME DESIGN

Afin de synthétiser nos réflexions sur les considérations techniques relatives au Serious Game Design, nous pouvons essayer d'analyser la manière dont elles répondent aux deux questions induites par notre problématique *(p.14)* :

1 QUELLES SONT LES SPECIFICITES TECHNIQUES DE LA CREATION DE SERIOUS GAMES ?

Alvarez précisait qu'il n'identifiait pas de différences entre Serious Game et jeu vidéo au niveau formel *(p.56)*. Notre étude des outils techniques de *« Game Design »* et de *« Serious Game Design »* semble effectivement confirmer sa conclusion. D'une part, **les outils techniques de « Game Design » et de « Serious Game Design » proposent globalement les mêmes fonctionnalités**. En effet, les outils de *« Serious Game Design »* sont ouvertement inspirés par leurs ancêtres que sont les « usines à jeux ». Par exemple, *SGTools (p.214)* reprend les bases de *Virtools (p.182)*, *e-Adventure (p.222)* s'inspire librement de *Adventure Game Studio (p.174)*, alors que *Thinking Worlds (p.214)* applique à la 3D temps réel des recettes éprouvées par des outils de création de jeux d'aventure *(p.174)*. Notre propre retour d'expérience sur l'évolution de *SCANeR* illustre également la manière dont les « usines à jeux » peuvent servir de source d'inspiration pour améliorer un outil de *« Serious Game Design » (p.215)*.

D'autre part, nous recensons de **nombreuses expérimentations réussies d'utilisation d'outils techniques de Game Design pour la création de Serious Games, ou pour la mise en place d'activités de création vidéoludique visant une finalité utilitaire**. Par exemple, du côté des *« usines à jeux »*, ***Céline Dunoyer*** utilise *Quandary* pour écrire des fictions interactives avec des élèves de 6e *(p.166)* ; ***Jacob Habgood*** s'appuie sur *Stagecast Creator* pour travailler avec des enfants de 7 ans dans un cadre extrascolaire *(p.180)* ; ***Mark Overmars*** enseigne l'informatique à ses étudiants d'université avec *Game Maker (p.179)* ; et nous avons fait réaliser un Serious Game à un étudiant en stage grâce à *The Games Factory 2 (p.195)*. En ce qui concerne le *« Jeu 2.0 »*, *Scratch (p.189)* et *Gamestar Mechanic (p.210)* ont été explicitement conçus pour être utilisés à des fins pédagogiques ; *LittleBigPlanet* est utilisé en classe de collège par ***Julien Llanas*** *(p.206)* et, avec *Spore*, est utilisé pour organiser des concours de création de Serious Games *(p.209)* ; le potentiel pédagogique de *Kodu* est évalué dans des écoles australiennes *(p.209)* ; tandis que *GameSalad* permet à des enfants en école primaire de créer leurs propres Serious Games avec le soutien de leurs instituteurs *(p.206)*. Nous pouvons également retenir les expérimentations de ***El-Nasr & al.*** sur l'utilisation des outils de *« modding »* à des fins pédagogiques, notamment pour l'apprentissage de l'informatique *(p.163)*.

Si les outils techniques de *« Serious Game Design »* ne sont pas fondamentalement différents de ceux de *« Game Design »*, nous observons cependant qu'ils proposent certaines **petites fonctionnalités supplémentaires** destinées à faciliter la création de Serious Games. Par exemple, les outils techniques de *« Serious Game Design »* que nous avons étudiés disposent

en standard des méthodes de « suivi du joueur » *(p.145)*. Que ce soit par l'intégration de la norme *SCORM* pour la diffusion du jeu à travers un *LMS (p.145)* ou le recours à des méthodes de suivi personnalisé *(p.145)*, ces outils facilitent grandement la mise en place du suivi des joueurs pour évaluer la pertinence des Serious Games réalisés. Nous remarquons également que certains outils techniques de *« Serious Game Design »* intègrent des outils théoriques afin de guider l'utilisateur dans son processus de conception. Par exemple, *<e-Adventure>* propose un langage graphique permettant de structurer le scénario du jeu *(p.225)*, alors que *Adventure Author* propose un « écran à idée » permettant d'assister le concepteur lors de la phase *« Imaginer » (p.226)*. De telles fonctionnalités sont, pour l'instant, absentes des outils techniques de *« Game Design »* que nous avons étudiés.

2 QUELLES SONT LES APPROCHES PERMETTANT DE FACILITER LA CREATION DE SERIOUS GAMES ?

Globalement, **les approches permettant de faciliter l'aspect technique de la création de Serious Games sont les mêmes que celles utilisées pour simplifier la création de jeu vidéo de divertissement.** Cela représente un avantage considérable pour le secteur du Serious Game, qui peut donc utiliser les nombreux outils techniques destinés aux amateurs du secteur du Game Design *(p.48)*. Existants depuis les années 1980, ces outils regroupent de nombreuses approches techniques de facilitation de la création vidéoludique *(p.166, p.177, p.214 et p.220)*. Les principales approches que nous avons identifiées sont :

- *La limitation du niveau technologique des jeux créables*. La première approche de facilitation de la création vidéoludique consiste tout simplement à limiter la complexité technologique des jeux qui sont réalisés. Cela impacte notamment le type de rendu graphique utilisé *(p.194)*. Par exemple, nous avons vu des outils spécialisés dans la création de jeux vidéo basés sur un rendu textuel *(p.166)*. La création de la partie graphique de ce type de jeu est bien plus aisée que la création de graphismes 2D, elle-même plus simple que la modélisation de graphismes 3D requise pour la création d'un jeu vidéo en 3D temps réel.

- *La limitation des composantes créables*. Avec le *modèle ISICO (p.159)*, nous avons mis en évidence de manière théorique l'existence de quatre « parties » qu'il faut créer lors de la réalisation d'un jeu vidéo :
  **« Etat Initial »* : l'état initial du système à état variable qu'est le jeu.
  **« Input »* : les moyens proposés au joueur pour envoyer des informations au système.
  **« Compute »* : les mécanismes internes qui permettent au système de changer d'état.
  **« Output »* : la manière dont le système communique son état courant au joueur.
  Nous constatons que, afin de faciliter la création d'un jeu vidéo, certains outils ne permettent pas à l'utilisateur de créer lui-même ces quatre composantes. Par exemple, un outil comme *Cartoon Network Game Creator (p.210)* ne permet de créer que l'*« Etat Initial »* du jeu, à travers la construction de niveaux. Les trois autres « parties » du jeu sont générées automatiquement par l'outil, simplifiant ainsi grandement la création de jeux. En contrepartie de sa grande facilité d'accès, ce type d'approche limite sévèrement la créativité des concepteurs.

- *Choix d'un outil spécialisé ou généraliste*. En se spécialisant dans un genre de jeu donné, les outils techniques peuvent proposer de nombreuses parties « préconstruites » qui réduisent considérablement le temps de réalisation d'un jeu vidéo *(p.176)*. Mais cette approche peut parfois limiter la créativité des utilisateurs, et n'est de toute façon pas envisageable pour des outils généralistes. Ces derniers misent plutôt sur l'accessibilité de leur interface pour simplifier la création vidéoludique *(p.177)*.

- *Le recours à des éditeurs de contenu émergeants ou à des menus de choix prédéfinis.* Il s'agit tout simplement des deux grands types « d'outils » proposés par les logiciels de création vidéoludique *(p.160)*. Les menus de choix prédéfinis sont plus simples d'utilisation, mais ils limitent la créativité aux bornes de l'imagination du concepteur de l'outil *(p.184)*. À l'inverse, les éditeurs émergeants permettent de créer du contenu qui n'aura pas forcément été anticipé par le concepteur de l'outil, au prix d'une manipulation un peu plus complexe *(p.168)*. Chaque outil technique utilise une combinaison des deux approches qui lui est propre. Ce choix influe directement sur l'équilibre entre la liberté de création et la simplicité d'utilisation de chaque logiciel.

- *Choix d'un éditeur visuel ou d'un langage de programmation pour les règles de jeu*. Les *« éditeurs visuels »* permettent de créer des programmes informatiques sans avoir recours à un langage de programmation. Ils sont donc bénéfiques pour l'accessibilité de l'outil, en particulier auprès d'un public novice *(p.180)*. D'un autre côté, les outils basés sur un langage de programmation sont un bon moyen d'apprentissage de l'informatique *(p.187)*. L'emploi d'un langage de programmation commun peut même constituer un tremplin vers le monde professionnel du jeu vidéo *(p.186)*.

En jouant sur ces différentes approches, les outils que nous avons analysés font preuve d'une grande variété en terme d'accessibilité et de possibilités de création. Cette diversité d'outils permet de faciliter l'aspect technique de la création de Serious Games pour une large variété d'utilisateurs. Mais une telle diversité pose également la difficile question du choix d'un outil adapté à un public donné. En effet, comme le précise ***El-Nasr & al.*** *(p.164)*, **le choix d'un outil technique adapté conditionne grandement la capacité des utilisateurs à créer un jeu vidéo ou un Serious Game**. Nous proposons alors un système de classification des outils techniques destiné à faciliter ce choix : le *modèle ISICO étendu* *(p.160)*. Les critères de ce modèle classificatoire synthétisent les différentes caractéristiques des outils techniques de création vidéoludique, permettant ainsi de choisir facilement un outil adapté. Lors de notre travail de recherche, nous avons classifié un corpus de 400 outils techniques avec cet outil, ce qui semble pour l'instant indiquer sa pertinence. Cette thèse présente des tableaux classifiant des outils techniques du *« Jeu 2.0 »* *(p.202)* et de *« Serious Game Design »* *(p.228)*. Nous avons également créé une base de donnée collaborative en ligne, qui permet de rechercher des outils techniques en utilisant les critères du *modèle ISICO étendu*. Réalisation concrète destinée à faciliter le choix d'un outil technique adapté à un public donné *(p.162)*, ce site est accessible à l'adresse suivante : http://creatools.gameclassification.com

3 SYNTHESE

L'étude des considérations techniques relatives au Serious Game Design semble confirmer l'hypothèse que nous avions émise suite à l'analyse de son aspect théorique *(p.154)* : **à ce jour, les approches permettant de faciliter la création de Serious Games sont principalement focalisées sur l'aspect technique**. Une des principales explication semble résider dans la possibilité d'utiliser les nombreux outils techniques du *« Game Design »* pour la création de Serious Games, alors que nous avons observé que les outils théoriques de *« Serious Game Design »* avaient tendance à se distinguer de ceux utilisés pour la création de jeux vidéo de divertissement *(p.152)*.

A la lumière de cette conclusion, il semble donc que les futures initiatives souhaitant apporter des approches de facilitation de la création de Serious Games doivent principalement se concentrer sur l'aspect théorique du *« Serious Game Design »*. Mais contrairement à l'aspect technique, la simplification de la dimension théorique de la conception de Serious Game n'est

pas obligée de reposer sur des « outils ». Nous présentons, dans la prochaine partie de cette thèse, un retour d'expérience sur la mise en place de cours destinés à sensibiliser des étudiants à la conception de Serious Games. Dans ce contexte particulier, la simplification de l'aspect théorique du *« Serious Game Design »* repose principalement sur le cours dispensé par l'enseignant, bien que les étudiants doivent toujours s'appuyer sur des outils techniques pour être en mesure de créer leur jeu. Ce contexte d'application précis sera donc l'occasion d'éprouver les nombreux outils techniques étudiés ici pour la création de Serious Games, ainsi que de réfléchir à des moyens permettant de palier à leurs éventuelles limites.

[PARTIE IV]
UNE APPLICATION PEDAGOGIQUE DU SERIOUS GAME DESIGN

L'étude du *« Serious Game Design »* menée jusqu'ici, autant pour des considérations théoriques *(p.58)* que techniques *(p.155)*, nous permet d'appréhender quelques particularités de la création de jeux vidéo à vocation utilitaire. Elle nous permet également d'identifier plusieurs approches permettant de faciliter la création de tels jeux *(p.152 et p.230)*. Ces approches s'appuient principalement sur des outils, qu'ils soient théoriques ou techniques.

Nous proposons alors, dans cette dernière partie de la thèse, d'évaluer la pertinence de ces approches de facilitation au sein d'un contexte d'application précis : des cours reposant sur la création de Serious Games. Ces cours, que nous dispensons à des étudiants, s'adressent à un public généralement novice en matière de création vidéoludique *(p.235)*. Cette expérimentation nous permettra tout d'abord de réaliser que le contexte est également un facteur permettant de faciliter la création de Serious Games *(p.255)*. Mais surtout, elle nous permettra d'identifier plusieurs limites aux approches de facilitation mises en place par les outils étudiés dans cette thèse *(p.248)*.

Cette application concrète de nos travaux de recherche nous invite donc à réfléchir aux solutions permettant de répondre à ces limites *(p.257)*. Cette réflexion ouvre une nouvelle problématique de recherche : la réalisation d'un outil technique implémentant des solutions aux limites identifiées. Nous avons commencé à y travailler à travers deux expérimentations logicielles *(p.266)*, dont les résultats sont suffisamment encourageants pour nous inviter à poursuivre ce travail au delà de la thèse *(p.297)*.

LE « SERIOUS GAME DESIGN » COMME METHODE PEDAGOGIQUE ?

Dans la troisième partie de cette thèse, nous avons évoqué plusieurs expérimentations de création de jeux vidéo en contexte pédagogique *(p.230)*. Que ce soit dans un cadre scolaire ou extrascolaire, des enseignants s'appuient sur la création de jeux vidéo ou de Serious Games pour proposer des activités pédagogiques à leur apprenants. Ce contexte nous semble alors particulièrement intéressant pour évaluer la pertinence des divers outils que nous avons identifiés dans cette thèse.

Nous reviendrons tout d'abord sur quelques références théoriques pour tenter de cerner le potentiel pédagogique de la création de Serious Games. Nous présenterons ensuite un retour d'expérience sur l'utilisation du *« Serious Game Design »* comme méthode pédagogique auprès d'étudiants âgés de 20 à 25 ans.

1 CADRE THEORIQUE : LE CONSTRUCTIONNISME

Dans son travail sur l'utilisation des jeux vidéo à des fins d'apprentissage, ***Kafai*** (2006) met en évidence une différence fondamentale entre deux types d'approches pédagogiques :

- **« L'instructionnisme »**, qui consiste à utiliser des jeux comme support d'apprentissage. Au sein d'une activité encadrée par un enseignant, les élèves jouent à des jeux diffusant un contenu pédagogique. Cette approche renvoie directement à toutes les considérations sur la création de Serious Games que nous avons étudiées jusqu'à présent. L'apprentissage s'appuyant principalement sur la pratique d'un jeu, sa pertinence en terme de conception est donc primordiale. Nous avons d'ailleurs vu qu'il existait plusieurs approches possibles à ce niveau *(p.106)*.

- **« Le constructionnisme »**, qui s'appuie sur la création de jeux comme support d'apprentissage. Les élèves sont ici invités à créer un jeu, dont la conception nécessitera l'acquisition de compétences ou de savoirs souhaités par l'enseignant. Si nos questionnements sur les particularités de la conception de Serious Game semblent également applicables à cette approche, le fait que les créateurs ne soient a priori pas spécialisés dans la création vidéoludique nous renvoie directement à notre problématique *(p.14)*.

Kafai précise que l'approche *« instructionniste »* est la plus répandue. De nombreux enseignants sont habitués à créer des « matériaux pédagogiques » à destination de leur élèves pour faciliter la transmission du savoir. Au-delà des particularités du support vidéoludique, le fait d'utiliser un jeu vidéo en tant que matériel pédagogique ne semble donc pas si éloigné du fait d'avoir recours à un quiz, un film documentaire ou des articles de journaux en contexte scolaire. En dépit du fait qu'elle bénéficie actuellement d'une attention moindre de la part des pédagogues, l'approche *« constructionniste »* semble posséder un potentiel pédagogique dépassant les limites des approches *« instructionnistes »*. Par exemple, ***Kafai*** précise qu'elle apparaît comme potentiellement plus adaptée à la prise en compte des différents styles d'apprentissage propres aux individus. L'élève étant mis en situation de création, il possède une plus grande liberté pour développer son propre rapport au contenu pédagogique, et donc

de choisir sa manière de l'assimiler. Afin d'étudier l'approche *« constructionniste »*, il nous faut d'abord nous interroger sur sa définition et ses origines.

1.1 LE CONSTRUCTIONNISME

Le constructionnisme a été principalement popularisé par un chercheur du nom de ***Seymour Papert***. Ce disciple de ***Jean Piaget*** *(un des principaux théoriciens du constructivisme)* définit son approche pédagogique de la manière suivante (Papert & Harel, 1991) :

> *« Il est facile de résumer le principe du constructionnisme en une simple phrase comme « apprendre par la création ». […] Mon petit jeu de mot entre « construction » et « constructionnisme » donne une idée des deux facettes du constructionnisme : une facette « sérieuse » et une autre plus « amusante ». La facette sérieuse sera d'ailleurs familière aux psychologues, qui reconnaîtront là une doctrine issue de la famille des théories cognitives dites « constructivistes ». Le constructionnisme (dont le N s'oppose au V du constructivisme) partage la vision constructiviste de l'apprentissage en tant que « construction d'un savoir structuré » quelle que soient les conditions d'apprentissage. Il lui ajoute ensuite l'idée que cet apprentissage est particulièrement efficace dans un contexte où l'apprenant est consciemment engagé dans la construction de quelque chose, qu'il s'agisse d'un château de sable ou d'une théorie de l'univers. »*[99]

Ainsi, ***Papert*** définit le constructionnisme comme une approche constructiviste *(p.106)* associée à un contexte de création. Afin de mettre sa théorie en application, ***Papert*** a inventé un langage de programmation informatique : le *LOGO*. Très simple à utiliser, ce langage est conçu pour être accessible aux novices, et plus particulièrement aux enfants *(p.188)*. L'approche de ***Papert*** s'avérant pertinente sur le terrain (Papert, 1993), elle a influencé de nombreux autres pédagogues. Si certains ont utilisé *LOGO* pour mettre leurs élèves en situation de création (Egenfeldt-Nielsen, 2006), d'autres se sont inspirés de la théorie de ***Papert*** mais ont développé leurs propres outils créatifs. Par exemple, ***Lavigne*** (2010) a développé un logiciel éducatif du nom de *POGO*. Il est destiné à faciliter l'apprentissage pour un public d'enfants souffrant de retards scolaires importants dus à divers handicaps *(troubles cognitifs, situation familiale difficile…)*. Ce programme permet aux enfants d'entrer une série de lettres *(« H » pour « Haut », « B » pour « Bas »…)* afin de déplacer un « crayon virtuel » à l'écran. Ce faisant, le logiciel permet de produire de nombreux dessins. Il s'agit là d'une adaptation des fonctions de dessin de *LOGO* et de sa célèbre tortue *(p.188)* aux besoins d'un public en difficulté. Ainsi, pour permettre aux enfants d'assimiler le fait que chaque lettre qu'il saisissent correspondra au dessin d'un trait à l'écran dans une direction précise, leur enseignant à tout d'abord mis en place un exercice « physique ». Des panneaux avec les différentes lettres ont été construits, puis donnés aux enfants. Un des enfants se tient alors debout devant ses camarades et joue le rôle du « crayon ». Lorsque les autres enfants lui montrent un des panneaux avec les lettres, il doit se déplacer dans la direction adéquate. Grâce à cette étape intermédiaire, les enfants ont pu prendre en main le logiciel et poursuivre leur processus d'apprentissage. Ce programme les a par exemple aidé à développer des facultés de

[99] *"It is easy enough to formulate simple catchy versions of the idea of constructionism; for example, thinking of it as "learning-by-making." […] My little play on the words construct and constructionism already hints at two of these multiple facets--one seemingly "serious" and one seemingly "playful." The serious facet will be familiar to psychologists as a tenet of the kindred, but less specific, family of psychological theories that call themselves contructivist. Constructionism--the N word as opposed to the V word--shares constructivism's connotation of learning as "building knowledge structures" irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe."*

planification de raisonnement *(en anticipant les mouvements produits par une série de lettres)*, ou encore à travailler le traçage des différentes lettres de l'alphabet *(certains de ces enfants éprouvant de grandes difficultés à ce niveau)*. Par la création de « dessins », ces enfants ont été acteurs d'un processus d'apprentissage. Leur enseignant a donc ici mis en place une activité pédagogique dont le support est un acte créatif. Cet exemple illustre l'approche générale du constructionnisme défini par ***Papert***. Mais cette définition mentionne également que le constructionnisme possède à la fois un côté « sérieux » et un côté « ludique ». Cela signifierait-il que le constructionnisme s'adapterai particulièrement bien aux Serious Games ?

1.2 CONSTRUCTIONNISME, JEUX VIDEO ET SERIOUS GAMES

Nous avons déjà évoqué de nombreux exemples d'approches constructionnistes s'appuyant sur la création de jeux vidéo. Un des thèmes régulièrement abordé par ce biais est l'apprentissage de l'informatique et de la programmation. Nous identifions par exemple les travaux de ***Overmars*** (2004) et ***Claypool*** (2005) s'appuyant sur *Game Maker (p.179)*, ou ceux de ***El-Nasr*** (2006) qui font appels aux outils de « modding » *(p.163)*. Les jeux vidéo peuvent également servir à améliorer des cursus universitaires, à l'image de l'expérimentation menée par ***Becker & al.*** (2007) pour instaurer la programmation de jeux vidéo comme activité pédagogique. Dans un registre similaire, ***Garcia-Mateos & al.*** (2009) ont imaginé un cursus de programmation en *C++* se déroulant sous forme d'une compétition. Du côté des matières littéraires, ***Dunoyer*** (2001) utilise *Quandary* pour travailler la rédaction de récits *(p.166)*, pendant que les chercheurs à l'origine des outils *Adventure Author* et *Flip* (Robertson & Howells, 2008) s'appuient sur la réalisation de « mods » pour aborder la création narrative *(p.99 et p.225)*. Dans un autre registre, certains enseignants choisissent d'enseigner les bases du *« Game Design »* à leurs élèves tout en leur permettant d'acquérir des compétences d'ordre général. Par exemple ***Llanas*** (2009) s'appuie sur *LittleBigPlanet* pour l'apprentissage du travail en équipe *(p.206)*, ***Clark & al.*** (2009; 2010) ont mis en place des ateliers visant à redonner confiance à des élèves en situation d'échec scolaire, *Kodu* a été utilisé dans des écoles australiennes *(p.209)*, alors que *Gamestar Mechanic* est au cœur du cursus d'une école primaire expérimentale *(p.210)*. Rappelons enfin que *Scratch (p.189)* est le support à des ateliers créatifs dans de nombreuses écoles à travers le monde (Peppler & Kafai, 2007).

Comme nous pouvons le voir, ces exemples d'approches constructionnistes s'appuyant sur la création vidéoludique à des fins éducatives dispensent souvent un apprentissage lié à l'informatique et aux nouvelles technologies. Dans une revue de littérature sur le sujet, ***Hayes & al.*** (2008) mettent ainsi en évidence que les expérimentations en la matière sont souvent destinées à l'apprentissage de disciplines techniques, au détriment d'autres domaines tels que la littérature, l'histoire, l'éducation civique… Ces chercheurs appellent donc les enseignants qui s'appuient sur le constructionnisme et le jeu vidéo à explorer plus profondément son potentiel pédagogique. Et justement, il semblerait qu'un des moyens permettant d'étendre cette approche à une large variété de thématiques consiste à **remplacer la création de jeux vidéo de divertissement par la création de Serious Games**.

Le travail fondateur en la matière est à n'en pas douter l'expérimentation menée par ***Kafai*** (1994) auprès d'enfants en école primaire. A raison d'une heure par jour et sur une durée de 6 mois, cette chercheuse a proposé des ateliers de création vidéoludique à un groupe de 16 élèves. Les jeux vidéo que ces enfants devaient créer n'étaient pas destinés au simple divertissement. L'objectif proposé à ces élèves de CM1 était de réaliser un jeu vidéo permettant l'apprentissage du concept mathématique de « fraction » à des élèves en CE2. Pour cela les enfants avaient accès à un ordinateur équipé du langage de programmation *LOGO*. L'expérimentation de ***Kafai*** est d'autant plus intéressante qu'elle a été menée avec plusieurs groupes-tests. Ainsi, dans cette école primaire suréquipée en ordinateurs grâce à un

programme pilote, quatre groupes d'élèves ont été constitués. Les deux premiers groupes, qui jouaient le rôle de groupes de contrôle, ont suivi des cours d'initiation à la programmation *LOGO* durant leur heure quotidienne d'atelier informatique. Un troisième groupe bénéficiait d'une forme d'apprentissage constructionniste à travers la réalisation des Serious Games éducatifs, encadrée par ***Kafai***. Enfin, un quatrième groupe était également impliqué dans une activité constructionniste à travers la réalisation non pas de Serious Games, mais de logiciels purement éducatifs traitant également du sujet des fractions. Globalement les deux groupes bénéficiant des ateliers constructionnistes ont fait preuve d'un apprentissage de meilleure qualité en ce qui concerne la maîtrise du concept de « fractions » comme pour l'apprentissage de la programmation informatique en *LOGO*. Si les progrès constatés pour les groupes ayant réalisés un Serious Game ou un logiciel pédagogique sont comparables, leurs créations sont par contre très différentes :

> *« Pour la création d'un logiciel pédagogique, les élèves ont principalement choisi l'approche « Dire et Montrer » pour introduire le concept de fraction à leurs utilisateurs. […] Les divers projets réalisés proposent un format similaire : l'écran affiche une ou plusieurs représentations de fractions, accompagnées par un texte explicatif ou une question à choix multiple. […] Pour la création des jeux pédagogiques, c'est plutôt l'invention d'un contexte de jeu intéressant et amusant qui prime. […] Les projets réalisés font preuve d'une grande variété thématique : les 16 créateurs de jeux ont créé 16 jeux très différents. […] L'association du sujet des fractions avec des jeux a d'ailleurs amené une réflexion sur la conception d'un programme avec deux objectifs disparates : l'apprentissage des fractions est un sujet que la plupart des élèves trouvent plutôt ennuyeux […], alors que le fait de jouer à des jeux est « pas comme l'école » - c'est « amusant ». […] Les élèves ont donc décidé de concentrer leurs efforts sur la création d'un jeu qui soit amusant, car ils ont compris qu'un jeu doit être amusant pour être captivant, et donc que pour rendre l'apprentissage des fractions intéressant il fallait que ce dernier ait lieu dans un contexte qui soit amusant. […] Les jeux créés par les élèves mettent en évidence à quel point il est difficile de trouver une idée de jeu qui soit à la fois centrée sur le concept de fractions tout en restant amusante. »*[100] *[p.269-274]*

Ainsi, en plus de perfectionner leur apprentissage du concept de fractions, les élèves du groupe ayant réalisé un Serious Game ont été sensibilisés aux problématiques de conception spécifiques à ce type de jeu *(p.106)*. En conclusion de son travail, ***Kafai*** note que l'approche constructionniste associée à la réalisation de Serious Games a permis aux élèves d'apprendre de nombreuses choses, aussi bien sur le contenu pédagogique qu'il doivent transmettre à travers leur jeu que sur la démarche de création en elle-même. Elle observe également que ce mode d'apprentissage, qui laisse une plus grande part à la créativité personnelle, permet de respecter les différents styles d'apprentissage propres à chacun des élèves. Au-delà de cette

[100] *"In the Instructional Software Design Project, the students took the tutorial or « Show and Tell » format as the most used format for designing their instructional software and introducing their learners to fractions. […] Across projects there was a similarity in format: The screen displayed one or more fraction representations, accompanied by either an explanatory text or a question to be answered. […] In The Game Design Project, creating an interesting or playful game context was the most dominant feature. […] There was a rich variety of game themes across all projects: 16 game designers in the project designed at least 16 different games. […] The combination of fractions with games also raised the issue of competing design issues: Learning Fractions is a subject that many students feel is rather boring […], whereas the playing of game sis something « not like school » - it is « fun ». […] They decided to invest their energies in the design of the game with the understanding that what makes a game exciting is when it is fun, and that learning fractions could only be fun in a fun context. […] The games designed by the students point out how difficult it is to find a game idea that on one hand is central to fractions but on the other hand is also fun."*

meilleure prise en compte des individualités, la chercheuse note que la création de jeux vidéo fut pour les élèves un moyen d'explorer plus profondément les différents modes de représentation des fractions. Enfin, ce projet a été un moteur fort pour l'apprentissage de la programmation informatique. Comparé aux approches exposées précédemment, il semble donc que le fait de remplacer la création de jeux vidéo par la création de Serious Games permette l'apprentissage d'un « contenu sérieux » défini par l'enseignant tout en conservant la possibilité de découvrir la programmation informatique. Si la création de jeux vidéo à des fins pédagogiques semble intéressante, la création de Serious Games semble dotée d'un potentiel encore plus grand par la possibilité de traiter une large variété de thématiques en plus des aspects liés à l'informatique. Pour autant, la mise en place de tels ateliers est fortement liée au choix de l'outil technique qui sera proposé aux apprenants. Parmi les prochaines étapes envisagées pour son travail, ***Kafai*** proposait de mettre en place un outil technique plus puissant que *LOGO*. En effet, plusieurs de ses élèves ont manifesté leur frustration face à l'impossibilité de réaliser certaines de leurs idées à cause des limites de cet outil. Par exemple, la nature « monothreadé » de *LOGO* rend impossible l'exécution de plusieurs routines en parallèle, ce qui empêche la réalisation de nombreux types de jeux. ***Kafai*** propose donc d'introduire des langages de programmation concurrente, voire des langages de programmation objet. Cela permettrait de sensibiliser les élèves à des concepts informatiques plus poussés tout en leur permettant de réaliser des Serious Games plus élaborés.

Cette expérience pionnière a inspirée d'autres enseignants pour la mise en place d'activités pédagogiques reposant sur la création de Serious Games. Une expérimentation faisant suite à celle de ***Kafai*** fut menée par ***Rieber & al.*** (1998) à travers le projet *KID DESIGNER*. En utilisant l'outil *Authorware* ***(Macromedia, 1992-2003)***, quatre classes de CM2 d'environ trente élèves ont créé un total de neuf Serious Games traitant de sujets allant de l'Histoire à la Physique en passant par les Mathématiques et la mythologie Grecque. Les conclusions des chercheurs à l'origine de ce projet rejoignent globalement celles de ***Kafai***, autant sur l'apport et l'intérêt pédagogique du constructionnisme que sur la nécessité d'avoir des outils techniques adaptés pour permettre aux élèves d'exprimer leur créativité. L'évolution technologique a donc bénéficié aux expérimentations plus récentes, à l'image du *projet Gameplay* mené dans des écoles primaires de la région parisienne avec l'outil *GameSalad* *(p.206)*. Chaque enseignant participant au projet a choisi une thématique en rapport avec ses cours *(astronomie, littérature…)* afin que ses élèves puissent réaliser un Serious Game sur le sujet. De son côté, ***Habgood*** (2005) utilise *Stagecast Creator* et *Game Maker* dans un club extrascolaire d'informatique pour permettre à des enfants de créer des jeux vidéo éducatifs sur différents thèmes *(p.180)*. Nous avons également évoqué la manière dont un enseignant avait utilisé *Virtuoso* pour que ses élèves améliorent un Serious Game qu'il avait commencé à créer *(p.220)*.

Au-delà des ateliers créatifs dans un cadre scolaire ou extrascolaire, une autre application du constructionnisme aux Serious Games se trouve dans l'organisation de concours de création. Nous avons déjà évoqué la *Digital Media and Learning Competition*, qui demandait aux participants de créer un niveau éducatif en utilisant les éditeurs des jeux *LittleBigPlanet* ou *Spore: Galactic Adventures* *(p.209)*. Un tel concours ouvert à tous mais limité à un seul outil technique a également été organisé par ***Orange*** (J. Alvarez & Maffiolo, 2011) à travers sa plateforme *WhoseGame* *(p.140)*. Il existe également des concours de création de Serious Games qui sont uniquement destinés à des élèves, à l'image du *National STEM Video Game Challenge*[101]. Sponsorisé par la ***Maison Blanche*** à l'initiative du président ***Obama***, ce concours de création s'adresse uniquement aux collégiens américains. L'objectif proposé est de réaliser un jeu vidéo traitant d'une thématique liée aux groupes de disciplines dites *STEM (Science, Technology, Engineering, Math)*. En effet, les écoles américaines constatent qu'elles

[101] Relevé le 26-06-11 à partir de http://www.stemchallenge.org/

ont de plus en plus de mal à attirer leurs étudiants vers ces filières, pourtant porteuses d'emplois. Le gouvernement américain a donc décidé d'organiser en 2010 un concours de création de Serious Games pour sensibiliser les jeunes américains à ces domaines. Ce concours propose de créer un jeu vidéo en utilisant soit *Gamestar Mechanic (p.210)*, soit *Scratch (p.189)*, soit *Game Maker (p.179)*, chaque outil correspondant à une catégorie différente de la compétition. Ce concours offre également une catégorie supplémentaire pour les adolescents qui utilisent un autre outil technique pour créer leur jeu, à partir d'une liste rassemblant *AgentSheets* ***(AgentSheets Inc., 1991-2010)***, *Flash (p.186)*, *GameSalad (p.202)*, *Kodu (p.187)*, *MicroWorlds* ***(LCSI, 1998-2009)***, *RPG Maker VX* et *RPG Maker XP (p.176)*, ainsi que *Stagecast Creator (p.180)*. Dans un autre registre, si la création de Serious Games peut être un moyen de sensibiliser des apprenants à une large variété de thèmes, elle semble également être potentiellement bénéfique pour l'industrie du jeu vidéo de divertissement. En effet, aux Etats-Unis la création vidéoludique n'est visiblement pas un métier attractif pour les femmes *(11% en 2006)* et les minorités ethniques *(4.5% en 2006)*, qui sont sous-représentées au sein des employés de ce secteur industriel. Afin de tenter de pallier ce problème, ***Argent & al.*** (2006) ont introduit la création de Serious Games au sein d'un cursus de « Game Design » visant à former des futurs professionnels de l'industrie du jeu vidéo de divertissement. En introduisant la création de jeux vidéo à vocation pédagogique, médicale ou traitant de sujets de société, les enseignants de ce cursus semblent avoir réussi à inciter de plus nombreux étudiants aux profils plus variés à s'orienter vers la création vidéoludique.

Ainsi, la création de jeux vidéo, et à plus forte raison de Serious Games, semble être un vecteur pédagogique particulièrement intéressant. Inspiré par les travaux que nous venons d'exposer, nous avons souhaité expérimenter une telle méthode pédagogique par nous-mêmes. Nous proposons donc de compléter ces considérations théoriques par un retour d'expérience sur cette approche auprès d'étudiants âgés de 20 à 25 ans.

2 RETOUR D'EXPERIENCE : ENSEIGNER PAR LE SERIOUS GAME DESIGN

Avec ***Julian Alvarez***, nous dispensons depuis 2006 des cours sur la conception de jeux vidéo aux étudiants de plusieurs cursus universitaires. L'objectif de nos cours est d'amener les étudiants à concevoir et réaliser un petit projet vidéoludique. A partir de 2007, nous avons remplacé la création de jeux vidéo de divertissement par la création de Serious Games (J. Alvarez, 2007). Si l'objectif pédagogique de nos cours reste de sensibiliser les étudiants aux problématiques de conception vidéoludique, le fait de s'appuyer sur le Serious Game présente à nos yeux plusieurs avantages. Tout d'abord, en plus de s'initier à la création de jeux vidéo, les étudiants enrichissent leurs connaissances sur d'autres sujets grâce aux recherches qu'ils font sur le « contenu sérieux » de leurs Serious Games. Ensuite, le fait d'imposer un sujet « sérieux » qui doit être traité à travers le jeu vidéo semble augmenter l'intérêt des étudiants pour le cours. En effet, et aussi surprenant que cela puisse paraître, ces étudiants qui ont grandi avec le jeu vidéo ne semblent pas avoir véritablement conscience du potentiel de ce support pour traiter de nombreux sujets *(cf. propos de Quentin p.195)*. Le fait de réaliser un Serious Game semble donc susciter chez eux un certain « conflit cognitif » quant à la possibilité d'aborder un sujet sérieux à travers un objet qu'il associent généralement au seul divertissement. Par rapport à l'objectif pédagogique de nos cours, le Serious Game s'avère donc plus pertinent que le jeu vidéo de divertissement, car il semble finalement amener les étudiants à réfléchir plus profondément aux problématiques liées à la conception vidéoludique.

Nos cours se déroulent selon plusieurs modalités. Certaines interventions ont lieu à raison d'une séance de trois heures par semaine durant un semestre, alors que d'autres sont concentrées sur une seule semaine à raison de sept heures de cours par jour. Lorsque nous disposons d'un volume horaire important, nous introduisons nos étudiants à des sujets connexes tels que l'histoire et la culture du jeu vidéo, ou les faisons réfléchir à des principes de « gameplay expérimental » *(p.140)*, avant de leur proposer de concevoir et réaliser leur propre Serious Game. Le déroulement de l'activité pédagogique que nous proposons à nos étudiants varie donc selon les années et les établissements. Néanmoins, elle s'appuie généralement sur la trame suivante :

1) Introduction

Les apprenants ignorant en général l'existence des Serious Games, nous introduisons notre cours par un exemple de jeu « marquant » afin de captiver leur attention et de susciter leur curiosité. Nous invitons donc un étudiant à « venir au tableau » pour jouer à un Serious Game traitant d'un sujet politique ou social important. Par exemple, *Darfur is Dying* ***(MTV, 2004)*** qui traite de la situation de crise humanitaire au ***Darfour***, et *September 12th* ***(Gonzalo Frasca, 2003)*** qui aborde la question de la réponse au terrorisme, suscitent toujours de nombreuses questions de la part des étudiants : *Qui a produit ce jeu ? Pourquoi faire un jeu sur ce thème ?* Le fait de répondre à ces questions nous permet d'introduire simplement la notion de *« Serious Game »*, tout en leur expliquant le déroulement du cours.

2) Découverte des Serious Games

Après cette courte introduction, nous invitons les étudiants à découvrir par eux-mêmes des exemples de Serious Games. L'objectif de cette phase est que les apprenants constatent l'immense variété de thèmes abordables à travers le jeu vidéo. De plus, les jeux qu'ils testent peuvent éventuellement les inspirer pour la création de leur propre Serious Game. Concrètement, nous invitons les étudiants à se connecter au site *Serious Games Opinions*[102]. Ce site propose une sélection d'une trentaine de Serious Games aux principes et sujets variés. Tous les titres proposés ont la particularité d'être jouables directement dans un navigateur Internet, ce qui permet d'éviter de subir les limitations techniques propres au réseau informatique des établissements *(pas d'installation sur les ordinateurs…)*. De plus, ce site *« Web 2.0 » (p.200)* permet à chaque utilisateur de noter et laisser des commentaires sur les titres testés. Ainsi, chaque étudiant testera quelques Serious Games puis laissera son commentaire d'évaluation directement sur le site. Ensuite, nous invitons chaque étudiant à « passer au tableau » pour présenter un des Serious Games qu'il a testé. D'autres apprenants dans la salle ayant testé le même jeu, un court échange s'engage alors sur chaque titre. Les étudiants discutent généralement des qualités ludiques de chaque Serious Game, ainsi que de sa pertinence dans la diffusion du contenu sérieux qu'il cherche à transmettre. Si besoin, l'enseignant donnera des informations complémentaires, car il connaît bien évidement chacun des jeux proposés sur le site. Une fois que chaque étudiant a présenté un jeu, nous essayons de proposer une courte synthèse sous la forme d'une présentation plus générale du phénomène des Serious Games : *dates clés de son histoire, importance économique, implantation dans la culture actuelle…* De plus, lorsque cette phase de découverte des Serious Game dispose d'un volume horaire important, nous proposons parfois aux étudiants d'endosser le rôle de « découvreurs ». A l'aide du site *Serious Game Classification*[103] *(p.32)*, ils doivent essayer d'identifier des Serious Games méconnus afin de les présenter à leurs camarades.

[102] http://www.seriousgamesopinions.org/
[103] http://serious.gameclassification.com/

3) Cours théorique sur les méthodologies et outils de conception vidéoludique

Nous dévoilons alors à nos étudiants la thématique qu'ils devront traiter à travers leur Serious Game. Suite à cela, nous leur proposons un petit cours sur les méthodologies et outils de conception vidéoludique adaptés au Serious Game. Nous nous basons pour cela sur notre propre méthodologie de conception dérivée du *modèle générique DICE* *(p.150)*. Nous leur proposons ensuite des outils et concepts qu'ils pourront utiliser pour chacune des étapes de cette méthodologie. Pour les étapes *« Définir »* et *« Imaginer »*, nous les introduisons aux notions importantes de *« Game Design »* telles que la structure d'un jeu *(p.68)*, les différents types de joueurs *(p.71)*, les différents modes d'association entre les dimensions « sérieuse » et « ludique » *(p.106)*… Pour les étapes *« Créer »* et *« Evaluer »*, nous leur proposons des outils techniques adaptés à la durée du cours et à leur niveau de compétence technique *(p.248)*. Dans le cas où les étudiants n'ont aucune compétence technique particulière, nous leur proposons d'utiliser une « usine à jeux » simple d'accès, comme *RPG Maker* *(p.251)* ou *The Games Factory 2* *(p.251)*. Les étudiants ne connaissant pas le logiciel, nous leur proposons alors un petit cours d'introduction à sa manipulation. En général, une séance de deux heures de présentation de l'usine à jeux est suffisante pour que des étudiants novices puissent commencer à l'utiliser pour créer leur propre Serious Game.

4) Conception et réalisation des Serious Games

Les étudiants se mettent alors en groupe, et commencent à travailler sur leur projet de Serious Game. Tout au long de cette phase, qui représente l'essentiel du cours en terme de volume horaire, l'enseignant quitte son rôle de « cours magistral » pour adopter une posture d'accompagnateur pédagogique. Alors que les apprenants travaillent à la réalisation de leur projet, l'enseignant devra assurer deux types de suivi :

- Le suivi *« théorique »* : l'enseignant devra accompagner les étudiants quant à la pertinence de leur jeu par rapport au sujet. Pour cela, il faut mettre en évidence les lacunes ou points positifs de leurs idées, et éventuellement les inciter à poursuivre leur recherche documentaire si le besoin s'en fait sentir. Il faut également vérifier la cohérence entre le sujet et le principe de jeu proposé, s'assurer qu'ils évaluent régulièrement leur Serious Game en le faisant tester à d'autres apprenants extérieurs au groupe… En résumé, l'enseignant guide les étudiants dans leur méthodologie de travail tout en essayant de veiller à la qualité des projets réalisés.
- Le suivi *« technique »* : si les apprenants n'ont aucune connaissance technique particulière, l'enseignant devra répondre à de nombreuses questions d'ordre pratique sur la manipulation de l'outil qu'ils découvrent durant le cours. Il est donc primordial que l'enseignant maîtrise bien le ou les outils techniques proposés. Dans le cas où les étudiants connaissent déjà parfaitement un outil technique, ce type de suivi est nettement moins important.

5) Présentation et évaluation des projets réalisés

Enfin, en guise d'évaluation, les étudiants sont amenés à présenter leur Serious Game au reste de la promotion. Après une courte présentation orale, un étudiant extérieur au groupe est désigné pour aller tester le jeu « au tableau ». Nous interrogeons alors cet étudiant sur ce qu'il a pensé du jeu, de manière à susciter un petit échange au sein de la promotion. Ces présentations donnent parfois lieu à de véritables débats de fond entre les étudiants *(p.245)*, et se révèlent donc particulièrement intéressantes d'un point de vue pédagogique. L'enseignant prendra néanmoins soin de récupérer les jeux et documents de travail produits par les apprenants afin de pouvoir leur attribuer une note. En général, nous notons les étudiants selon deux critères : la pertinence du concept de jeu proposé par rapport au sujet traité, et la qualité du Serious Game réalisé.

En suivant cette trame générale, nous avons pu proposer à de nombreux étudiants une activité pédagogique reposant sur la création d'un Serious Game. Les jeux ainsi créés par nos étudiants sont détaillés ci-après.

2.2 EXEMPLES DE REALISATIONS DES ETUDIANTS

Lors de nos différentes interventions, plusieurs cas de figure se sont présentés. Pour certaines formations, le temps alloué au cours n'était pas suffisant pour permettre aux étudiants de réaliser un Serious Game. Dans ces rares cas, nous avons simplement proposé aux étudiants d'imaginer un concept de jeu et d'en rédiger le *« Game Design Document »* *(p.44)*. Mais pour la majorité des interventions de ***Julian Alvarez*** ou de moi-même, les étudiants ont pu réaliser un petit Serious Game, par groupes de quatre ou cinq élèves en moyenne. Le tableau ci-dessous référence les types de formations, les thématiques proposées et le nombre de projets réalisés par nos étudiants depuis 2007. Au total, cela représente 61 Serious Games :

Tableau 15. Listes des thématiques des Serious Games réalisés par nos étudiants

Niveau	Thématique proposée	Nombre de jeux réalisés
Licence 3 en multimédia	Présenter un métier au choix pour aider des collégiens dans leur orientation scolaire	6
Master 1 en multimédia	L'hygiène alimentaire	3
	Les technologies de l'information et la communication	4
4e et 5e année en école d'ingénieur	L'hygiène alimentaire	4
	Téléphone portable et sécurité lors des soirées étudiantes	5
	Les technologies de l'information et la communication	5
	Présenter le métier d'ingénieur aux collégiens pour leur orientation scolaire	5
	Comment réussir ses études d'ingénieur	4
	Réaliser un jeu sur un sujet d'actualité *(« NewsGame »)*	14
Master 2 en informatique	L'hygiène alimentaire	3
	Les technologies de l'information et la communication	2
	Aide à l'insertion professionnelle des étudiants (trouver son premier emploi)	3
	Sensibilisation à la sécurité dans un environnement de travail	3

2.2.1 DES SERIOUS GAMES TRAITANT DE SUJETS FAMILIERS AUX ETUDIANTS

Certaines des thématiques proposées à nos étudiants sont volontairement liées à leur formation ou à leurs connaissances préalables. L'idée est de leur permettre de créer un jeu sur un sujet « sérieux » qu'ils maîtrisent bien. Par exemple, nous avons proposé à nos étudiants en école d'ingénieur de réaliser un jeu vidéo relatif à leur métier. L'objectif proposé est d'arriver à créer un jeu à destination d'un public de collégiens pour les aider dans leur orientation scolaire. Ainsi, le jeu *Make'em Chat* ***(2007)*** présente les bases du métier d'ingénieur réseau. Le joueur doit arriver à relier différents ordinateurs entre eux en évitant de surcharger son architecture réseau. Très amusant, ce jeu illustre de manière humoristique le quotidien d'un ingénieur réseau. De son côté, *Ingénioland* ***(2007)*** se déroule dans un univers peuplé d'elfes, de trolls et autres créatures issues de l'imaginaire *« médiéval-fantastique »*. Ce jeu de rôle propose au joueur une quête épique dans laquelle les différentes compétences du métier d'ingénieur *(recrutement et gestion d'un équipe, planification de projets, sens de la logique…)* seront nécessaire pour triompher.

88. Les projets étudiants *Make'em Chat* et *Ingénioland*

Dans un autre registre, nous avons également proposé aux étudiants de réaliser des Serious Games de type *« fiction interactive »* sur un sujet qui les touche directement : la prévention des risques liés aux soirées étudiantes. Le sujet précisait même que le téléphone portable devait avoir une place importante dans l'histoire, son rôle exact étant laissé à l'imagination des étudiants. Ces derniers ont alors produit des jeux aux tons très différents. Par exemple, *Ange ou démon ?* ***(2009)*** propose diverses situations réalistes avec des choix binaires. Face à un ami qui a manifestement trop bu, lui confisquez-vous ses clés ou restez-vous à rigoler en le voyant tituber jusqu'à sa voiture ? De son côté, *En soirée, oublie pas ta sécurité !* ***(2009)*** joue la carte de l'humour acide afin de « choquer » les joueurs pour essayer de les faire réfléchir sur les véritables dangers liés à la surconsommation d'alcool en soirée *(accident de voiture, coma éthylique, rapports sexuels non protégés…)*. Ce groupe d'étudiants a même été jusqu'à détourner des panneaux du code de la route pour proposer une petite « morale » aux différentes fins de cette histoire interactive.

89. Les projets étudiants *Ange ou démon ?* et *En soirée, oublie pas ta sécurité !*

La familiarité des apprenants avec les sujets proposés n'est pas toujours liée à leur statut d'étudiants. Par exemple, le sujet traitant des « technologies de l'information et la communication » est plus général, même si par leur âge et leur milieu social ils y sont forcément plus sensibles. Les Serious Games réalisés sur ce thème ont été soumis au concours de création de Serious Games organisé par ***Orange*** à travers sa filiale *WhoseGame* *(p.237)*. Certains groupes d'étudiants ont choisi d'aborder ce sujet par un angle ouvertement militant. Ainsi, *Orange Metallique* ***(2009)*** propose d'éviter l'apparition de tumeurs cérébrales chez les habitants d'une ville en luttant activement contre la prolifération des antennes relais. Pour cela, il faut littéralement tuer les agents de la société ***Orange*** qui passent dans la rue. Si un agent arriver à passer en esquivant les tirs du joueur, il plantera une antenne relais supplémentaire sur les toits de la ville. De par son contenu, il semblait évident que ce Serious

Game ne laisserait pas indifférent la société organisatrice du concours, elle-même opérateur de téléphonie mobile. Et effectivement, ce jeu n'a pas été sans poser de nombreuses questions aux organisateurs du concours (J. Alvarez & Maffiolo, 2011). Au final le jeu a été disqualifié car il ne respectait pas une clause du concours qui interdisait de citer une marque commerciale. Mais il a néanmoins eu un impact très fort au sein de la société ***Orange***. De son côté, le Serious Game *Quizz Telecom* ***(2009)***, un jeu de question-réponse simulant une discussion par messagerie instantanée, a remporté le prix du public pour la catégorie *« enrichir ses connaissances sur les télécommunications »*.

90. Les projets étudiants *Orange Métallique* et *Quizz Telecom*

2.2.2 DES SERIOUS GAMES TRAITANT DE SUJETS DE SOCIETE

Nous avons également tenté de proposer aux étudiants des thématiques d'ordre plus général. Les étudiants n'étant plus « experts » du sujet qu'ils doivent traiter à travers leur Serious Game, ils doivent cette fois-ci effectuer un travail de recherche documentaire sur le thème qu'il leur a été proposé. Ainsi, plusieurs promotions d'étudiants se sont vues confier la mission de réaliser un jeu vidéo traitant de l'hygiène alimentaire, avec pour finalité de sensibiliser un public de collégiens à cette question importante. Afin de pouvoir toucher un public différent de leur âge, nous avons demandé aux étudiants de réaliser un travail d'enquête préliminaire auprès de collégiens. Les étudiants sont donc allés dans des collèges pour interroger les élèves sur leurs loisirs et centres d'intérêt. Il est ressorti de ces études que les adolescents étaient friands de séries télévisées et de jeux vidéo. Certains étudiants ont donc décidé de s'appuyer sur les franchises les plus populaires, d'après leur enquête, pour réaliser un Serious Game. Ainsi, *La bouffe pour les nuls* ***(2008)*** reprend l'univers de la série *Les Simpsons* ***(Matt Groening, 1989-2011)***. Chaque joueur incarne un des membres de la célèbre famille dans une sorte de « jeu de l'oie » en 3D. Un tour de plateau représente une journée complète, qui est découpée en activités *(aller à l'école, faire du sport…)* correspondant à autant de cases sur un parcours à choix multiples. Ce parcours dispose également de cases « obligatoires » qui correspondent aux trois repas d'une journée. Selon les choix d'activités et d'aliments que font les joueurs, le bilan nutritionnel de leur personnage à la fin du parcours sera plus ou moins positif. De son côté, *Pakumon* ***(2008)*** détourne la série de jeux vidéo *Pokémon* ***(Nintendo, 1996-2011)***. Comme dans le jeu de rôle originel, les joueurs capturent des monstres qu'ils utilisent ensuite pour combattre et résoudre des quêtes. Sauf qu'ici, les joueurs doivent en plus s'occuper de nourrir leurs créatures. Et attention à ne pas les nourrir n'importe comment : seule une alimentation bien équilibrée permet aux monstres d'être efficaces au combat !

91. Les projets étudiants *La bouffe pour les nuls* et *Pakumon*

Au-delà de l'hygiène alimentaire, nous avons également proposé à nos étudiants de réaliser un *« NewsGame »*, autrement dit un Serious Game traitant d'un sujet d'actualité (Sicart, 2009). De notre point de vue, cette thématique présente un grand intérêt pédagogique. Tout d'abord, l'actualité recouvre de nombreuses thématiques différentes, ce qui permet aux étudiants d'opérer un choix du sujet auxquels ils sont les plus sensibles. De plus, grâce aux nombreuses ressources journalistiques disponibles sur Internet, les étudiants ont à leur disposition une riche documentation sur l'actualité qu'ils choisiront de traiter. Enfin, cette thématique nous semble permettre d'étendre l'horizon de nos cours au-delà de la seule culture vidéoludique. Concrètement, le fait de réaliser un jeu vidéo sur un sujet de société amène souvent les étudiants à avoir des débats de fond sur l'actualité. Par exemple, l'épidémie de la *« grippe A »* a beaucoup fait discuter les étudiants entre eux. Certains ont donc choisi d'en faire le sujet de leur Serious Game, à l'image de *Superflu* ***(2010)***. Le groupe d'étudiants à l'origine de ce jeu pensait que le seul moyen d'enrayer la pandémie était une collaboration à l'échelle mondiale, et ce malgré les différences de moyens financiers entre les pays. Ainsi leur Serious Game se joue en réseau, chaque joueur se voyant affecter une partie du monde qui abrite des villes et des usines de production de vaccins et d'antigrippaux. Les joueurs peuvent choisir d'envoyer les médicaments produits dans leurs usines où ils le veulent, dans leurs villes comme dans celles des autres joueurs. Le jeu a été sciemment conçu pour que le seul moyen d'enrayer la pandémie soit que tous les joueurs distribuent l'intégralité de leur production de médicaments aux premiers foyers d'infection, même si ces villes ne sont pas sur leur territoire. Ce Serious Game transmet donc la vision d'un groupe d'étudiants sur ce thème. Mais tous n'ont pas forcément la même vision d'une actualité suite à leur recherche documentaire. Ainsi, un autre groupe de la même promotion a réalisé le jeu *Flucorp Inc.* ***(2010)*** qui adopte une vision radicalement différente du problème. Ici le joueur incarne le dirigeant d'une société de production pharmaceutique. Le joueur commercialise différents produits *(équipements de protection, vaccins, anti-viraux…)* plus ou moins efficaces selon son taux d'investissement dans la recherche médicale. Si le joueur investi massivement ses fonds dans la recherche, il contribuera à endiguer rapidement l'épidémie, mais n'en tirera pas un grand profit financier. Alors qu'au contraire, s'il choisi sciemment d'investir « modestement » dans la recherche, il aura largement le temps de vendre plusieurs versions d'un même produit *(par exemple des vaccins de plus en plus efficaces…)*. Ainsi, le joueur est incité à ne pas « faire tout son possible » pour guérir l'épidémie s'il souhaite maximiser ses profits. Les étudiants ont ainsi un traitement quasi-militant de l'actualité, leur jeu critiquant indirectement la posture des laboratoires durant cette crise. Dans un autre registre, notons que *Superflu* illustre si besoin est qu'un Serious Game peut tout à fait être plébiscité pour ses qualités ludiques. Ainsi, les étudiants ayant réalisé ce jeu l'ont diffusé sur le réseau interne de leur école, où il rencontra apparemment un certain succès. Nous avons donc eu la bonne surprise d'accueillir, lors du cours de l'année suivante dans le même établissement, un groupe de quatre étudiants qui avait

choisi d'assister à notre cours car ils avaient pu jouer à *Superflu*. Cela leur avait visiblement donné envie de réaliser leur propre jeu vidéo.

92. Les projets étudiants *Superflu* et *Flucorp Inc.*

Enfin certains *« NewsGames »* réalisés par nos étudiants s'attaquent à des sujets d'actualité tellement sensibles qu'ils peuvent parfois en être choquants. Ainsi, *Tunisian Oppression* ***(2011)*** traite de la révolution Tunisienne selon le point de vue du dictateur ***Ben-Ali***. Plusieurs petits jeux d'action narrent l'épopée du dictateur. Tout d'abord, en incarnant sa garde, le joueur doit essayer de contenir la foule en utilisant des tirs à balle réelle. Le jeu étant conçu de manière à ce que le peuple finisse toujours par gagner, le joueur devra alors essayer de s'enfuir avec un maximum d'or avant de chercher un pays qui daigne l'accueillir. De part son aspect satirique, ce jeu peut heurter quelques sensibilités. Lorsque les étudiants ont présenté leur création à leurs camarades, un véritable petit débat s'est engagé sur le fond de cette actualité ainsi que sur la teneur du message véhiculé par ce jeu. Bien que les créateurs du Serious Game aient voulu montrer que *« le peuple gagnera toujours en dépit de l'acharnement du dictateur »*, le fait d'échanger avec leurs camarades leur a montré qu'il peut exister un certain écart entre le message voulu par le concepteur et sa réception par les utilisateurs. Cela fut encore plus flagrant avec le jeu *Escape from Port-au-Prince* ***(2010)*** qui traite d'un sujet particulièrement sensible : le terrible tremblement de terre qui a frappé ***Haïti*** en janvier 2010. Le joueur incarne un survivant de la catastrophe, et doit essayer de quitter la ville tout en restant en vie. Le groupe d'étudiant à l'origine de ce Serious Game a volontairement adopté un traitement très « cru » de l'actualité. Durant son périple, le joueur croisera donc des fosses remplies de cadavres, devra faire des choix cornéliens pour se procurer de la nourriture, et surtout se battre pour la conserver. Le tout sous la pression constante de la faim qui gagne l'avatar du joueur et menace de le tuer. Lors de la présentation de ce projet à la fin de notre cours, plusieurs étudiants ont été littéralement choqués. Au-delà de la catastrophe en elle-même, c'est véritablement son traitement qui a été questionné. En effet, bien qu'ils soient amateurs de jeux vidéo et aient eux-mêmes réalisés un Serious Game sur l'actualité, certains étudiants ont trouvé que *« ce sujet est trop grave pour être traité à travers un jeu vidéo »*. Au-delà de la question de la violence du traitement, c'est véritablement le fait que *« cette violence est vraie pour des gens dans le monde, à l'inverse de celle que l'on trouve dans les jeux vidéo de divertissement »* qui a heurté certains étudiants. A l'inverse, d'autres étudiants ont défendu la légitimité du support vidéoludique pour traiter tout type de sujets, au même titre que les autres supports. Ainsi, et de manière complètement spontanée, un débat d'une trentaine de minutes a fait suite à la projection de ce Serious Game. En plus de tout ce qu'ils ont pu apprendre sur la conception de jeu vidéo et sur les sujets d'actualité qu'ils ont traité, tous les étudiants de cette promotion ont donc été amenés à réfléchir sur les possibilités et limites du jeu vidéo en tant que moyen d'expression.

93. Les projets étudiants *Tunisian Oppression* et *Escape from Port-au-Prince*

Au final, nous constatons à la lumière des différents jeux vidéo réalisés par nos étudiants que le recours à la conception de Serious Games comme méthode pédagogique peut permettre à des apprenants de réfléchir à des questions qui dépassent largement le cadre de la création vidéoludique. Cette approche associant constructionnisme et Serious Games semble donc intéressante à des fins pédagogiques. Cependant, pour permettre à ces étudiants de réaliser des Serious Games alors qu'ils n'ont jamais créé de jeu vidéo auparavant, il faut leur proposer un cadre adapté. Nous avons déjà évoqué le plan d'activité pédagogique que nous avons mis en place *(p.241)*. Une autre question est primordiale pour mener ce type d'activité : le choix des outils techniques. Selon le niveau de compétences des étudiants, la durée du cours ou encore le type de matériel informatique disponible, différents choix s'offrent à l'enseignant. Il est primordial que l'outil permette aux apprenants de réaliser leurs idées, tout en restant accessible à leur niveau de compétence technique. Sur les cinq années durant lesquelles nous avons dispensé ce type de cours, nous avons pu évaluer plusieurs outils.

2.3 DES OUTILS DE GAME DESIGN POUR LA PEDAGOGIE CONSTRUCTIONNISTE

Bien que nos étudiants possèdent forcément des compétences plus développées que les jeunes élèves suivis par ***Kafai***, ***Rieber*** ou ***Habgood*** *(p.237)*, tous n'ont pas forcément reçu de formation en création multimédia ou en informatique. C'est particulièrement le cas de nos étudiants issus d'école d'ingénieur en génie civil et en génie chimique. S'ils savent bien utiliser l'outil informatique, ils n'ont généralement aucun savoir particulier relatif à la création multimédia. Pour ces étudiants « novices », nous devons donc proposer des outils particulièrement simples d'accès, d'autant plus que notre module de cours est relativement court. A l'inverse, nos étudiants issus d'un master en informatique maîtrisent parfaitement la programmation et sont donc en mesure d'utiliser des outils aussi puissants que complexes. Entre ces deux extrêmes, nous avons aussi des étudiants issus de cursus orientés multimédia. Si ces derniers maîtrisent avec aisance la production d'éléments graphiques et sonore, il a parfois fallu leur proposer des outils techniques adaptés à leur niveau « intermédiaire » en programmation informatique. Au final, nous avons donc proposé à nos étudiants des outils allant d'applications bureautiques standard à des langages de programmation professionnels en passant bien évidement par des usines à jeux *(p.166)*. Voici les retours que nous avons pu observer lors de l'utilisation de ces quelques outils par nos étudiants.

2.3.1 LES OUTILS THEORIQUES

Evoquons tout d'abord le cas des outils théoriques. Nous avons rapidement pu constater que les outils théoriques inventés par les Game Designers professionnels *(p.59)* étaient trop complexes pour nos étudiants. Nous leur avons donc proposé une approche plus simple, en la forme du tableau *« Objectifs / Moyens / Contraintes »* évoqué précédemment *(p.87)*. Nous

avons pu constater sur le terrain que certains groupes utilisaient ce tableau pour formaliser leurs idées lors des premières phases de réflexion. Au-delà de ce modeste guide créatif, nous n'avons pas encore trouvé de moyen véritablement pertinent pour introduire la richesse des outils théoriques de Game Design au sein de nos cours. Cette question reste donc ouverte, même si l'idée d'intégrer un outil théorique directement à l'intérieur d'un outil technique semble être une voie prometteuse *(p.222 et p.225)*.

2.3.2 POWERPOINT

Dans nos cours qui mettent l'accent sur la dimension « histoire interactive », la grande majorité des étudiants ne possèdent aucune compétence technique particulière. Pour la plupart d'entre eux, la maîtrise d'un outil de création s'arrête aux outils bureautiques tels que *Powerpoint* ***(Microsoft, 1987-2010)***. La faible durée de ce module a généralement conduit nos étudiants à utiliser cet outil pour créer leurs Serious Games de type « fiction interactive ». Simple d'accès et déjà connu des étudiants, *Powerpoint* possède l'avantage d'être particulièrement adapté à la réalisation de jeux d'aventure simples, sous forme de « romans photos » interactifs. Au-delà de l'impossibilité de réaliser un autre genre de jeu avec cet outil, nos étudiants ont signalé quelques lacunes techniques diverses. Leur principal souci vient du fait que les joueurs peuvent facilement *« passer à l'écran suivant »*. A la base, *Powerpoint* est conçu pour faire défiler des diapositives de manière linéaire. Or leurs histoires interactives ont une structure non-linéaire, ce qui est parfois mal géré par cet outil. Travaillant en groupe, ils apprécient cependant la possibilité de réaliser séparément les diverses « branches » d'une histoire avant de les réunir dans un fichier commun. Le travail de groupe en parallèle est donc facilité par cet outil, qui a permis à nos étudiants de réaliser *Ange ou démon ?* ainsi que *En soirée, oublie pas ta sécurité !* *(p.243)*.

2.3.3 FLASH

Flash *(p.186)* est à ce jour le logiciel que nos étudiants ont le plus utilisé. Cela s'explique par plusieurs raisons. Tout d'abord, les étudiants des filières « multimédia » connaissent déjà bien cet outil, ce qui facilite son emploi pour créer des Serious Games. Ensuite, pour l'année où nous avons proposé à nos étudiants d'envoyer leurs réalisations au concours organisé par *Whosegame* *(p.243)*, les règles du concours stipulaient que seuls des jeux réalisés avec *Flash* pouvaient être soumis. Enfin, cette technologie étant particulièrement populaire pour la création de jeux vidéo *(p.186)*, certains de nos étudiants souhaitent tout simplement apprendre à l'utiliser. Ils profitent alors du projet à réaliser durant notre cours pour découvrir ce nouvel outil technique. Pour autant, même si *Flash* est réputé pour sa facilité d'utilisation, il ne s'agit pas d'un logiciel qui est particulièrement accessible aux novices. Si la création d'animations avec cet outil est relativement simple à apprendre, la maîtrise de son langage de script propriétaire, l'*ActionScript*, demande une certaine pratique. La dernière version du langage, *l'ActionScript 3.0*, s'apparente même, en terme de complexité et de puissance, aux langages de programmation communs, et plus particulièrement à *Java (langage orienté objet structuré de la même manière, présence d'un « Garbage Collector », classes de base similaires…)*. Nous avons donc mis en place plusieurs stratégies pour faciliter l'apprentissage de *Flash* à nos étudiants. Tout d'abord, nous leur fournissons au début de chaque cours un grand nombre de « moteurs de jeu » open-source réalisés en *Flash*. Ainsi, s'ils veulent par exemple réaliser un jeu de plateforme ou de course, ils n'ont pas à programmer l'intégralité du jeu à partir de zéro. Ils peuvent s'appuyer sur une base existante pour aller plus vite dans la réalisation de leur Serious Game. De même, tous les projets que nos étudiants créent sont ensuite mis en ligne de manière open-source dans une base de données. Ainsi, les étudiants de chaque promotion peuvent étudier comment les promotions précédentes ont réalisées leurs jeux. Un de nos groupes d'étudiants a même directement utilisé un jeu réalisé par une promotion précédente comme base de son Serious Game. Ces stratégies ont aidé à la création de projets tels que *Make'em Chat*, *Orange Métallique*, *Quizz Telecom*, *Pakumon* et *Flucorp Inc*. *(p.243)*.

Signalons également que *Flash* dispose d'un mode de programmation de type *« éditeur visuel »* en plus de son *« langage de script propriétaire »*. Concrètement, cet outil propose un écran dans lequel l'utilisateur peut créer des scripts simples sans avoir à les écrire. Pour cela, il lui suffit de sélectionner diverses instructions du langage à partir d'un menu déroulant. Un panneau s'affichera ensuite avec différents champs permettant d'éditer facilement les *« paramètres »* des instructions ajoutées au script. Ce mode *« assistant de script »* est néanmoins très limité dans les choix qu'il offre : *jeu d'instructions partiel, impossibilité de créer de nouvelles variables ou de leur affecter des valeurs…* Il s'avère pourtant très pratique pour initier des novices aux bases de la programmation informatique. Ainsi, bien que cela ne soit pas directement lié aux Serious Games, nous dispensons également des cours de « création numérique » à des étudiants en Arts Plastiques et Arts Appliqués *(niveaux Licence 3 et Master 1)*. Ces étudiants, qui maîtrisent de nombreuses techniques plastiques allant du dessin à la sculpture en passant par la photographie et la peinture, n'ont en général jamais utilisé d'outil de création multimédia, et ne possèdent aucune connaissance de la programmation informatique. Pourtant, en utilisant le mode *« assistant de script »* de *Flash*, nous avons réussi à leur enseigner les rudiments de la programmation informatique en quelques heures. Certains étudiants ont alors pu s'affranchir du mode *« assistant de script »* pour programmer leurs créations directement et sans assistance. Ce cours de « création numérique » suit également une approche pédagogique constructionniste *(p.235)*, l'objectif étant ici la création d'une œuvre artistique au lieu d'un Serious Game. Rien n'empêche donc a priori d'utiliser cette même approche pour des cours de création de Serious Games, même si à ce jour le besoin ne s'en est pas présenté. En effet, pour nos cours de « création numérique », les étudiants réalisent des projets individuels. Ils sont donc tous obligés de maîtriser l'outil technique. Alors que pour les cours s'appuyant sur les Serious Games, plusieurs apprenants travaillent sur un même projet. Pour les groupes de quatre ou cinq étudiants créant un Serious Game avec *Flash*, nous observons généralement la même situation. Un ou deux étudiants visiblement plus à l'aise avec l'outil endossent le rôle de « programmeur » et réalisent donc la majeure partie du travail technique, tandis que les autres membres du groupe se consacrent à d'autres tâches *(réalisation de graphismes, de sons, écriture de dialogues…)*. Au-delà de la possibilité de travailler séparément la programmation de la réalisation des graphismes, *Flash* s'accorde d'ailleurs assez mal avec le travail de groupe en parallèle. Les seules exceptions sont les projets constitués de plusieurs « mini-jeux », qui permettent à deux « programmeurs » de travailler chacun sur un jeu différent. Mais si le Serious Game repose sur un concept unique, une seule personne à la fois pourra y travailler dessus en utilisant *Flash*.

2.3.4 LANGAGES DE PROGRAMMATION COMMUNS

Lorsque nos étudiants maîtrisent déjà la programmation informatique, ils choisissent généralement de s'appuyer sur un langage de programmation commun. Au sein des projets réalisés par nos étudiants, le *Java* arrive en tête, suivi par le *C++*, le *PHP* et le *C#*. Les étudiants peuvent alors réaliser des jeux techniquement plus évolués, à l'image de *La bouffe pour les nuls* et son moteur 3D réalisé pour l'occasion, ou de *Superflu* et son mode de jeu en réseau *(p.243)*. Ces étudiants utilisent parfois des frameworks spécialisés dans la création vidéoludique, à l'image de *XNA* ***(Microsoft, 2006-2010)*** qui permet de programmer des Serious Games tournant aussi bien sur *PC (Windows)* que sur *Xbox 360* à l'aide du langage *C#*.

De tels frameworks, ou les usines à jeux s'appuyant sur des langages de programmation commun, permettent à des utilisateurs expérimentés de réaliser des jeux vidéo relativement élaborés en un temps record. Dans le cadre de notre *CIFRE* au sein du centre culturel ***Odyssud***, et plus particulièrement au sein de sa ludothèque, nous avons été amené participer à l'organisation d'un concours de création vidéoludique. La *Global Game Jam*[104] est une

[104] http://globalgamejam.org/

compétition internationale de création de jeux vidéo se déroulant sur un week-end entier. A travers le monde, plusieurs « lieux d'accueil » sont ouverts et accueillent de nombreux participants. Ceux-ci forment alors des équipes, et disposent de 48h pour créer un jeu vidéo à partir d'un thème donné. L'édition 2011, pour laquelle le centre culturel ***Odyssud*** a participé en tant que « lieu d'accueil », a réuni 6500 participants à travers le monde grâce aux 170 « lieux d'accueils » répartis dans 44 pays. Au total, près de 1500 jeux vidéo auront été créés par les participants. En ce qui concerne l'antenne organisée par la ***ludothèque*** et l'***E.C.M.*** du centre culturel ***Odyssud***, nous avons modestement accueilli 17 participants qui ont constitué deux équipes différentes. Une de ces équipes, composée de 10 membres, a conçu et réalisé un petit Serious Game en 48h à l'aide de *Unity* *(p.186)*. Une fois le concept du jeu défini et validé par toute l'équipe, il ne leur a pas fallu plus de 30 heures pour réaliser le jeu *Celestial Equilibrium* ***(2011)***, destiné à sensibiliser les joueurs à l'écologie. Ces professionnels du multimédia étaient certes habitués à travailler avec *Unity*. Mais le fait d'arriver à réaliser un jeu d'action en 3D temps réel, à partir de zéro, en moins de 30h, n'aurait clairement pas été envisageable sans le recours à un outil technique simplifiant et accélérant les phases de réalisation d'un jeu vidéo. S'ils représentent la solution la plus puissante d'un point de vue créatif, les outils techniques s'appuyant sur un langage de programmation commun nécessitent de bonnes bases en programmation informatique afin d'être utilisés. Ils sont donc difficilement envisageables pour des novices en la matière, à plus forte raison dans le cadre d'une activité pédagogique à durée très courte telle que la création d'un Serious Game en quelques jours.

2.3.5 RPG MAKER

De nombreux apprenants, qui n'ont aucune connaissance en programmation informatique, souhaitent quand même réaliser des Serious Games plus élaborés que les « romans photos interactifs ». Ils doivent alors se tourner vers des usines à jeux s'appuyant sur une approche *« éditeur visuel »* pour la création des règles de jeux *(p.183)*. Plusieurs groupes de nos étudiants, qui souhaitaient réaliser un jeu de rôle ou d'aventure, ont employé le logiciel *RPG Maker (versions XP et VX)* *(p.176)*. Le fait que cet outil soit livré avec des librairies de graphismes « prêt à l'emploi » ainsi qu'un *« éditeur de niveau »* très intuitif représente un énorme avantage pour nos étudiants. De même, la nature « spécialisée » de cette usine à jeux implique qu'elle automatise la création des principales règles de jeu caractérisant un jeu de rôle. Durant la phase de réalisation, les étudiants peuvent donc se concentrer sur l'implémentation du scénario afin de réaliser un Serious Game particulièrement généreux en contenu. Par exemple, *Ingénioland* propose plus d'une demi-heure de jeu captivante alors qu'il a été réalisé en seulement cinq jours, tandis que *Escape from Port-au-Prince* propose un univers de jeu plus réduit mais riche en interactions *(p.243)*. Principal défaut signalé par nos étudiants, *RPG Maker* s'accorde très mal du travail en équipe. En effet, lors de la réalisation technique d'un Serious Game, une seule personne à la fois peut travailler sur le projet. Une autre de leurs remarques porte sur le fait que, pour réaliser des interactions plus complexes, *« l'éditeur visuel »* de règles proposé par cette usine à jeux ne suffit plus. Il faut alors recourir au langage de script intégré, qui est loin d'être aussi simple d'accès. Nos étudiants se contentent généralement de « copier-coller » des scripts disponibles sur les nombreux sites Internet fournissant de l'aide sur *RPG Maker*. Malgré ces quelques défauts, *RPG Maker* reste un outil particulièrement intéressant pour la création de Serious Games du genre *« jeu de rôle et d'aventure »* par des apprenants sans compétence technique particulière.

2.3.6 THE GAMES FACTORY 2

Pour des étudiants ne sachant pas programmer mais se sentant bridés par les limitations des usines à jeux spécialisées *(p.166)*, *The Games Factory 2* *(p.178)* semble un outil technique particulièrement adapté. Il s'agit d'une usine à jeux généraliste, qui permet donc de réaliser tout type de jeu vidéo pour peu qu'il soit en 2D. Ensuite, et contrairement à *RPG Maker*, il s'appuie uniquement sur un *« éditeur visuel »* pour la création des règles de jeu. Aucun

langage de script n'étant proposé pour réaliser les règles les plus complexes, cet éditeur permet de réaliser des systèmes de règles de jeu particulièrement riches sans avoir à apprendre de langage de programmation. Si de nombreux outils disposent d'un *« éditeur visuel »* pour la partie *« Compute »* *(p.193)*, rares sont ceux qui en proposent un permettant d'utiliser l'ensemble des fonctionnalités de l'outil. *The Games Factory 2* est également fourni avec un logiciel de dessin et d'animation intégré, ce qui permet aux étudiants de réaliser l'intégralité de leur projet au sein d'un seul outil, un peu comme *Flash*. Mais à la différence de ce dernier, *The Games Factory 2* est adapté au travail en parallèle dans un groupe. Chaque étudiant peut travailler séparément sur un « niveau » ou un « écran » du jeu, puis facilement rassembler le tout dans un seul fichier par la suite. Lors de l'introduction de cette usine à jeux dans nos cours, nous avons pour la première fois constaté que, dans un groupe réunissant quatre ou cinq étudiants, ces derniers s'essayaient à tous les rôles. Là où l'utilisation de *Flash* tendait à concentrer le travail technique sur un ou deux membres du groupe, ici le fait de ne pas avoir à apprendre de langage de programmation permet à tous les membres du groupe de travailler sur la partie *« Compute »* du jeu *(p.160)*. De plus, et contrairement aux autres outils que nous avons testés, il n'a pas été nécessaire de proposer aux étudiants une base « open-source » pour réaliser leur jeu. En partant de zéro, tous les groupes d'étudiants ont réussi à réaliser un Serious Game relativement riche en seulement quelques jours, à l'image de *Tunisian Oppression* *(p.243)*.

Pour autant, cet outil technique n'est pas exempt de défauts. Ainsi, la famille d'usines à jeux à laquelle appartient *The Games Factory 2* a toujours proposé des « comportements standards » afin d'accélérer la réalisation de jeux vidéo. Par exemple, il est possible d'attribuer, d'un simple clic, un mouvement de type « jeu de plateforme » à tout objet présent dans un niveau. Dans la première version de *The Games Factory* ***(1998)***, ces comportements souffrent de nombreuses limitations. Par exemple, il est impossible de modifier séparément les vélocités en X et en Y des mouvements. Ou encore, pour les mouvements de type « voiture de course », il est n'est pas possible de configurer précisément la manière dont la voiture rebondit. Ces mouvements sont même parfois *« buggés »*, une « voiture de course » ayant par exemple tendance à s'accrocher aux obstacles du décor. Malheureusement, bien que publiée huit ans plus tard, la seconde version de ce logiciel ne corrige pas ces limitations. Les étudiants doivent donc essayer tant bien que mal de palier aux problèmes de ces comportements livrés en standard. Un de nos étudiants, qui avait quelques bases en programmation mais dont le groupe avait choisi d'utiliser *The Games Factory 2* pour que tout le monde puisse participer à la réalisation des règles du jeu, nous a confié que *« si on avait accès au code, au moins il serait possible de modifier les comportements proprement au lieu d'essayer de bricoler pour contourner les limites des moteurs standards »*, avant de nuancer ses propos *« mais bon c'est aussi vrai que si on avait programmé le jeu, à ce moment du projet on en serait encore à réaliser des routines graphiques »*. Malgré cet aspect « bricolage » qui peut parfois être bloquant pour les étudiants, *The Games Factory 2* présente selon nous un autre avantage d'un point de vue pédagogique. Le fait que cette usine à jeux soit « généraliste » implique qu'aucune règle par défaut n'est préprogrammée dans les jeux qu'elle permet de créer. Les étudiants doivent donc tout réaliser par eux-mêmes, ce qui leur permet de découvrir les nombreuses « astuces » inventées par les développeurs de jeux vidéo. Par exemple, nos étudiants étant jusqu'à alors uniquement des joueurs, ils étaient persuadés que les collisions entre les objets d'un jeu vidéo sont détectées directement entre lesdits objets. Or, pour des questions d'optimisation, les collisions sont rarement détectées directement entre les objets mais plutôt entre des « boites de collisions » possédant des formes géométriques simples. En ayant à programmer explicitement toutes les règles de leur Serious Game, les étudiants sont amenés à découvrir de nombreuses techniques de ce genre. Dans une veine similaire, un de nos étudiants a pu s'initier à la programmation d'intelligence artificielle pour réaliser des ennemis qui chassent le joueur dans un labyrinthe, pour son Serious Game au principe proche de *Pac-man*. Ainsi, malgré le fait que cet outil ne s'appuie pas sur un langage de

programmation, il permet tout à fait de travailler l'algorithmique avec les apprenants qui l'utilisent. *The Games Factory 2* nous semble donc constituer un choix très pertinent pour nos cours, en particulier pour les apprenants ne sachant pas programmer mais qui souhaitent réaliser des Serious Games sans être restreint à un genre vidéoludique particulier.

2.3.7 GAME MAKER ET CONSTRUCT

Signalons également qu'un groupe d'étudiants a réalisé un Serious Game en utilisant *Game Maker* *(p.179)*, que l'on peut voir comme une bonne alternative à *The Games Factory 2*. Ce groupe d'étudiants n'avait aucune connaissance particulière en programmation, mais ils ont réussi à prendre en main le logiciel lors d'un cours s'étalant sur une seule semaine. Ils ont néanmoins dû se baser sur un moteur de jeu open-source afin d'être en mesure de créer leur Serious Game dans le temps imparti. Notons qu'un autre groupe d'étudiants avait essayé de réaliser un Serious Game en utilisant le logiciel *Construct*, un clone open-source de *The Games Factory 2* *(p.178)*. L'instabilité de l'outil à l'époque *(novembre 2009)* les a néanmoins conduit à se rabattre sur *RPG Maker* pour réaliser leur projet.

2.3.8 OUTILS DE CREATION GRAPHIQUE ET SONORE

Enfin, au delà de la réalisation d'un programme informatique matérialisant les règles du jeu vidéo, les étudiants doivent la plupart du temps produire des nombreuses images, sons et musiques pour l'habiller. A l'exception des filières multimédia et artistiques dont les étudiants sont plus que compétents dans le domaine, les capacités créatives des étudiants à ce niveau sont souvent modestes. La plupart du temps, ils se contentent prendre des photos ou vidéos, voire de récupérer des images, sons et musiques sur Internet. Nous constatons néanmoins que lorsque l'outil technique utilisé incorpore un outil de dessin, à l'image de *The Games Factory 2* et *Flash*, cela tend à inciter les étudiants à essayer de dessiner par eux-mêmes au lieu de récupérer des images externes. Même les étudiants ne se sentant pas une âme d'artiste essayent alors de modifier les images qu'ils récupèrent, s'investissant ainsi un peu plus personnellement dans l'aspect graphique de leur Serious Game. Mais les étudiants peuvent aussi avoir recours à d'autres logiciels comme source d'images. Ainsi, un de nos groupes d'étudiants était « rebuté » par la production graphique, mais souhaitait néanmoins créer un jeu visuellement cohérent. Sur les bons conseils de ***Julian Alvarez***, ils ont alors utilisé le programme *Lego Digital Designer* ***(Lego, 2004-2010)***. Ce logiciel, distribué gratuitement par le fabricant de jouets ***Lego***, permet de créer des modèles 3D en utilisant une représentation virtuelle des briques du célèbre jeu de construction. A la base, ce service permet d'inventer son propre modèle ***Lego***, pour ensuite commander au fabricant les pièces nécessaires à sa construction. Mais ce programme permet également de prendre des captures d'écrans des modèles construits par l'utilisateur. Nos étudiants ont donc utilisé ce logiciel pour construire les différents éléments graphiques de leur jeu *(maisons, personnages, meubles…)*. Ils les ont ensuite exportés sous forme d'images afin des les intégrer à leur projet. Au-delà de ces petites astuces pour les aspects graphiques et sonores de leur Serious Games, les étudiants nous suggèrent chaque année de créer une sorte de « banque commune » de graphismes et sons qu'ils pourraient utiliser pour leurs projets. Si nous n'avons pas encore mis en place une telle fonctionnalité, le recours à une approche « Jeu 2.0 » *(p.200)* semblerait tout à fait répondre à cette demande.

2.3.9 GUIDER LES APPRENANTS DANS LA PRISE EN MAIN D'UN OUTIL

En plus de ces retours d'expérience sur l'utilisation de plusieurs outils techniques par nos étudiants, notons ici une remarque générale qui peut sembler « naturelle » mais qu'il est facile d'oublier une fois sur le terrain. En tant qu'enseignant, si l'on souhaite que nos apprenants utilisent un outil technique qu'ils ne connaissent pas, il faut absolument les guider durant la découverte dudit outil. En effet, nous avons fait une fois l'erreur de simplement proposer une liste d'usines à jeux aux étudiants *(en leur précisant les points forts et faibles de chaque usine à jeux)*. Nous espérions naïvement qu'ils choisiraient par eux-mêmes un logiciel adapté à leur

niveau, et nous poseraient par la suite des questions sur son utilisation. Face à l'échec de cette approche, nous proposons dorénavant une seule usine à jeux aux étudiants « novices » *(en estimant par nous-même leur niveau de compétence en informatique)*, et les guidons dans la prise en main du logiciel sous forme d'un cours *(p.241)*. Cette seconde approche s'est révélée bien plus pertinente pour amener des apprenants sans compétence informatique particulière à réaliser des Serious Games relativement élaborés *(p.243)*.

3 SYNTHESE : LE CONTEXTE POUR FACILITER LE SERIOUS GAME DESIGN

Que ce soit d'après la lecture d'études scientifiques *(p.235)* ou suite à notre propre retour d'expérience *(p.240)*, le recours à la création de Serious Games comme support d'activité pédagogique apparaît comme une approche particulièrement intéressante. En impliquant l'apprenant dans la réalisation d'un projet, elle permet de le sensibiliser aussi bien aux problématiques de conception d'un Serious Game qu'au « contenu sérieux » que leur jeu devra transmettre.

Nous constatons également que cette application pédagogique du *« Serious Game Design »* constitue une approche permettant de le faciliter. En effet, la présence d'un pédagogue pour accompagner et guider les apprenants dans leur processus de création de Serious Games participe activement à l'adaptation du *« Serious Game Design »* à leur niveau de compétences. Par exemple, tel que nous l'avons expérimenté durant nos propres cours, un « médiateur humain » sera bien plus efficace qu'un « outil théorique » pour accompagner la réflexion créative des apprenants *(p.241)*. **La mise à disposition d'outils adaptés n'est donc pas la seule approche permettant de faciliter le « Serious Game Design » : le fait d'avoir un référent spécialiste du sujet pour accompagner les concepteurs représente également une approche très pertinente.**

Pour autant, la mise en place d'une telle activité pédagogique s'accompagne de plusieurs conditions. Tout d'abord, il semble évident qu'il faut avant tout que l'enseignant soit intéressé par cette approche et maîtrise parfaitement la thématique sérieuse qu'il propose à ses apprenants. Mais il faut également que l'enseignant soit en mesure d'accompagner ces derniers dans la manipulation des différents outils qu'ils utiliseront pour créer leurs Serious Games. Si les bases de l'aspect théorique de la conception de Serious Games peuvent toujours être enseignées par le pédagogue, la question de l'outil technique est loin d'être triviale. En effet, le choix d'un outil technique doit répondre à de nombreuses contraintes :

- L'âge et le niveau de compétence technique des apprenants : *maîtrise préalable d'un outil ? Connaissance d'un langage de programmation ? Âge des apprenants pour la compréhension de certains concepts ?…*
- La durée de l'activité pédagogique : *prévoir des outils simples d'accès si les apprenants doivent les découvrir durant un module de cours de faible durée…*
- Le matériel à disposition de l'enseignant : *nombre et puissance des ordinateurs à disposition, modalité d'accès à Internet, possibilité d'installer des logiciels sur les machines de l'établissement…*
- Les fonctionnalités de l'outil selon les projets envisagés : *les apprenants vont-il faire des jeux en 2D ou en 3D ? Ont-ils besoin de créer tous les aspects de leur jeu ? Se concentrent-ils sur un genre vidéoludique précis ou non ?…*

Les réponses à ces différentes questions permettent généralement de choisir, parmi les nombreux outils techniques existants *(p.166, p.200 et p.214)*, celui qui sera le plus adapté aux besoins de l'enseignant et de ses apprenants. Cependant, malgré la richesse des outils disponibles, il peut parfois arriver que l'outil adapté à un contexte donné n'existe pas encore. Dans ces cas-là, l'enseignant essaiera alors de sélectionner le ou les outils qui s'approchent au mieux de ses attentes. Mais une autre solution est envisageable : tenter d'inventer un nouvel outil technique adapté à ces besoins spécifiques. Etant donné le caractère informatique de notre travail de recherche, c'est cette dernière approche qui motive le dernier chapitre de cette thèse. Nous y tenterons d'imaginer et réaliser un outil technique qui soit parfaitement adapté aux besoins d'apprenants lors de cours similaires à ceux présentés dans ce chapitre.

GENESE D'UN OUTIL TECHNIQUE DE SERIOUS GAME DESIGN

Comme nous l'avons évoqué à plusieurs reprises dans cette thèse *(p.228 et p.231)*, la diversité des outils techniques existants est une des approches permettant de faciliter la création de Serious Games pour une large variété d'utilisateurs. Concrètement, cette facilitation de la création de Serious Games présente des intérêts dans plusieurs domaines. Tout d'abord au sein de l'industrie du Serious Game. Cela permet d'une part aux concepteurs professionnels d'utiliser des outils de conception et de prototypage entraînant un gain de productivité, ainsi que d'impliquer plus facilement les « experts » ou les commanditaires impliqués dans leurs projets *(p.214)*. Cette facilitation du Serious Game Design présente également un intérêt certain pour l'éducation. Par exemple, elle peut permettre à des enseignants de créer des « matériaux pédagogiques » captivant l'attention de leurs élèves *(p.220)*. Mais il est également possible d'impliquer directement ces derniers dans une pédagogie constructionniste reposant sur la création de Serious Games comme support d'apprentissage *(p.235)*. Au-delà des aspects théoriques, un des principaux piliers de toutes ces applications du *« Serious Game Design »* est l'outil technique qui sera proposé aux utilisateurs. Celui-ci doit être adapté aussi bien aux désirs créatifs de l'utilisateur qu'à son niveau de connaissance technique, tout en prenant en compte les objectifs industriels *(productivité)* ou éducatifs *(apprentissage)* du contexte d'utilisation.

Cette diversité des contextes d'utilisation a tout d'abord motivé un travail de recherche et d'analyse visant à référencer et classifier de nombreux outils techniques *(p.159)*. Mais malgré la diversité des outils regroupés dans cette base de donnée, il peut exister des contextes d'utilisation spécifiques pour lesquels aucun outil existant ne répond véritablement à tous les besoins. Ainsi, lors de nos propres cours s'appuyant sur la création de Serious Games *(p.240)*, nous avons pu tester plusieurs outils. Si les résultats constatés sont encourageants au vu des productions des étudiants *(p.243)*, nous avons quand même pu observer de nombreuses limitations pour chacun de ces outils *(p.248)*. Malgré leurs nombreuses qualités respectives, aucun d'entre eux ne semble pour l'instant être parfaitement adapté au contexte d'utilisation spécifique de nos cours.

Nous souhaitons donc consacrer le dernier chapitre de cette thèse à la création d'un nouvel outil technique de Serious Game Design pensé pour ce contexte d'utilisation précis. Dans un premier temps, nous essaierons de revenir de manière synthétique sur les fonctionnalités les plus pertinentes des nombreux outils étudiés dans cette thèse. Nous complèterons cette synthèse par des références à d'autres travaux de recherche en cours sur ce thème, ainsi que par le retour d'expérience de nos étudiants. Ces diverses sources permettrons d'établir un « cahier des charges » des fonctionnalités qui semblent pertinentes pour ce nouvel outil. Nous exposerons ensuite deux projets distincts visant à réaliser un logiciel répondant à ce cahier des charges. Si le premier projet n'a pas abouti, le second a permis de réaliser un premier « moteur » fonctionnel répondant à quelques aspects de ce cahier des charges. Cette base semble suffisamment prometteuse pour espérer continuer le développement de cet outil au-delà du contexte de cette thèse.

1 DEFINITION D'UN « CAHIER DES CHARGES »

Nous proposons ici un ensemble de choix de conception et de fonctionnalités qui nous semblent pertinentes pour la création d'un outil destiné à des activités d'enseignement par le Serious Game Design. Ces choix sont motivés par les conclusions observées dans d'autres travaux de recherches sur le sujet, par le retour d'expérience formulé par nos étudiants ainsi que par l'analyse des nombreux outils techniques étudiés dans cette thèse.

1.1 DISSOCIER « MOTEUR D'INTERPRETATION » ET « INTERFACE AUTEUR »

Notre première considération porte sur la relation entre l'outil technique en lui-même et les jeux qu'il permet de créer. La quasi-totalité des outils que nous avons étudiés dans cette thèse fonctionnent d'une manière similaire. Tout d'abord, une « interface auteur » permet de créer le jeu, dont les données seront ensuite sauvegardées sous forme d'un fichier. Afin de pouvoir distribuer le jeu une fois qu'il est terminé, tous les logiciels disposent ensuite d'un « moteur d'interprétation » qui permet de transformer ce fichier de données en jeu fonctionnel. Concrètement, ce « moteur d'interprétation » est une sous partie de « l'interface auteur » qui contient uniquement les lignes de code permettant de convertir les « choix » du concepteur en un jeu vidéo. Le jeu ainsi distribué est plus « léger » que l'outil technique car il n'inclut pas le code lié au fonctionnement de « l'interface auteur ». Au-delà des considérations purement techniques, ce choix permet d'échelonner le développement de l'outil. Il est ainsi possible de commencer par créer le « moteur d'interprétation » avant d'entamer la réalisation de « l'interface auteur ». *<e-Adventure>* a par exemple été développé selon ce principe, la première version du logiciel impliquant de rédiger des fichiers *XML* à la main pour créer un jeu, l'interface avec « éditeur visuel » n'étant apparue qu'après *(p.222)*. De plus, ce choix permet également d'envisager l'existence de plusieurs « interfaces auteur » liées à un même « moteur d'interprétation ». Cela permettrait de proposer, à terme, plusieurs approches de création vidéoludique *(p.258)*. *Flash* est par exemple basé sur ce principe *(p.186)*. Il est possible de créer des jeux avec ce logiciel soit avec un simple éditeur de texte, soit par le biais d'une interface auteur proposant plusieurs « éditeurs visuels » plus intuitifs. Notons cependant qu'en dissociant ainsi le « moteur d'interprétation » de « l'interface auteur », le jeu résultant est un programme « interprété » et non « compilé ». Si cela impacte négativement sa performance, il s'agit d'une condition nécessaire pour permettre de modifier un jeu « à la volée » à des fins de prototypage *(p.264)*. Nous dissocierons donc le « moteur d'interprétation » et « l'interface auteur » de notre outil technique.

1.2 UN LOGICIEL S'APPUYANT SUR LE MODELE « OBJET »

Si plusieurs paradigmes de programmation existent (Roy & Haridi, 2004), force est de constater que la vaste majorité des outils techniques de création vidéoludique s'appuient sur le paradigme *« impératif »*, et plus particulièrement sur le modèle *« Orienté Objet »* *(p.156 et p.214)*. Ce choix se comprend tout à fait suite à la lecture de travaux sur la nature du jeu *(p.156)*. D'après ces publications scientifiques, un jeu peut-être vu comme un *« système à état variable »*. Ce *« système »* est composé de plusieurs *« éléments »* qui sont eux-mêmes à *« état variable »*. Le *« système »* possède également un ensemble de *« règles »* qui régissent les changements d'état de chacun de ses éléments. Ce type de modélisation générale du « jeu » semble intellectuellement très proche des principes à la base du modèle *« Orienté Objet »*. Ainsi, le fait d'avoir à créer un jeu vidéo comme étant composé « d'objets », au sens informatique du terme, permet de conserver la structure en *« éléments »* proposée par les définitions scientifiques du « jeu » *(p.156)*. Les différentes « propriétés » d'un « objet » informatique permettent alors de refléter les *« changements d'états »* de chacun des *« éléments »* du jeu. Quant aux *« règles »* du jeu, il semble facile de les implémenter au cœur

des « méthodes » des différents classes « d'objets » composant le programme informatique. Nous souhaitons donc nous appuyer sur le modèle *« Orienté Objet »* pour réaliser notre outil technique.

Avant d'aborder plus précisément le type d'architecture choisie *(p.258)*, revenons sur une des limitations rencontrées par nos étudiants avec le logiciel *The Games Factory 2* *(p.251)*. Bien que cet outil repose sur une approche inspirée par le modèle *« Orienté Objet »*, il ne permet pas au concepteur de créer des objets qui se référencent entre eux. Cela peut s'avérer problématique lorsque plusieurs instances d'une même classe d'objet sont présentes dans un niveau. Par exemple, les étudiants ayant travaillé sur *Tunisian Oppression* *(p.245)* ont réalisé un niveau dans lequel le joueur doit tirer sur des manifestants. Afin de pouvoir gérer l'effet de perspective *(déplacement en profondeur des manifestants)* tout en détectant leurs collisions avec le décor, les étudiants ont du utiliser une « boite de collision ». Pour cela, ils ont créés des petits objets ronds *(invisibles)* qui se déplacent en respectant la perspective. Chacun de ces objets ronds est ensuite associé un autre objet représentant le manifestant à proprement parler. Ce type de technique, couramment utilisée pour programmer un jeu de ce genre, fait partie des « astuces » que nos étudiants découvrent en réalisant leur premier jeu vidéo *(p.251)*. Pourtant, dans ce cas précis, ils n'ont pas pu implémenter cette technique. En effet, *The Games Factory 2* ne permet la création de règles qu'entre classes ou groupes d'objets. Il n'est donc pas possible d'associer chacun des objets de la classe « manifestant » avec un, et un seul, des objets de la classe « petit objet rond ». Cette limitation est du à l'absence d'un quelconque moyen de créer une référence à un objet dans les différents types de variables proposés par le logiciel. De même, lorsque le concepteur crée une règle *(par exemple des tests de positions)*, il ne lui est pas possible de l'appliquer uniquement à une instance précise d'une classe d'objet. Loin d'être anecdotique, cette limitation technique empêche souvent les étudiants de réaliser leurs idées. Les limitations sont encore plus importantes avec les logiciels qui ne gèrent pas les variables de type « scalaire » *(tableaux, vecteurs…)*, comme c'est par exemple le cas de *Game Develop* *(p.178)*. Aussi simple que soit son interface pour le *« Compute »* *(p.260)*, notre outil technique devra donc gérer aussi bien les références précises à une instance dans les règles et entre les objets, ainsi que proposer la gestion des variables scalaires.

1.3 UNE ARCHITECTURE DE TYPE « OBJETS CONTENEURS DE DONNEES »

Lors de notre étude des usines à jeux, nous avons constaté qu'il existe deux grandes approches d'utilisation du modèle *« Orienté Objet »* pour représenter l'architecture d'un jeu vidéo *(p.198)*. Si tous les outils dissocient la structure ludique en un ensemble « d'objets » représentant les *« éléments »* du jeu, la manière dont ils implémentent les *« règles de jeu »* diffère fondamentalement. D'un côté, nous avons une approche que nous avons baptisée « objets intelligents ». Respectant profondément le modèle *« Orienté Objet »*, chaque *« élément »* du jeu contient toutes les règles qui s'appliquent à lui-même. Ces « comportements » sont tous simplement des *« règles de jeu »* qui sont implémentées à travers les « méthodes » de l'objet informatique représentant *« l'élément de jeu »*. *Game Maker* est par exemple basé sur cette approche, ce qui le rend pertinent pour l'initiation à la programmation *« Orienté Objet »* *(p.179)*. De l'autre côté, nous avons des outils dont les *« règles de jeu »* sont complètement dissociées des « objets » qui représentent les *« éléments de jeu »*. Ici les « objets » informatiques représentant les *« éléments de jeu »* ne possèdent aucune « méthode » propre, mais uniquement des « propriétés » représentant leur état. Les *« règles de jeu »* sont donc implémentées dans une autre classe « d'objet informatique ». Nous avons appelé cette approche « objets conteneurs de données », notamment illustrée par *The Games Factory 2* *(p.178)*. Cette approche implique une gestion plus pointue de la manière dont les *« règles »* sélectionnent les objets auxquelles elles s'appliquent, ce qui peut parfois introduire certaines limites, à l'image de celles évoquées précédemment *(p.257)*. Au

premier abord, cette seconde approche semble moins pertinente. De plus, elle s'éloigne du modèle *« Orienté Objet »*, ce qui la rend plus difficile à implémenter. Et pourtant c'est bien cette approche « objets conteneurs de données » qui est la plus pertinente pour la conception de jeu, car c'est celle qui se rapproche le plus de la manière dont pense un Game Designer. ***Pierce*** l'illustre parfaitement dans un de ses articles sur le sujet (Pierce, 2011). En se référant au cas des jeux de plateau, il précise par exemple qu'un pion du jeu *Monopoly* ***(Charles Darrow, 1935)*** n'incorpore pas les règles qui conditionnent son déplacement. Ces dernières sont regroupées de manière claire dans un autre « objet » : le livret de règle. A partir de cet exemple, il note que les Game Designers n'imaginent pas les jeux avec des *« éléments »* incorporant les *« règles de jeu »*, mais qu'ils ont plutôt tendance à voir ces deux entités comme séparées lorsqu'ils rédigent le *« Game Design Document »* *(p.44)*. C'est ensuite le développeur qui fera le choix d'utiliser les « méthodes » des objets pour les implémenter les « règles de jeu » au sein des « éléments ». ***Pierce*** affirme même que l'approche *« Orienté Objet »* à tendance à « parasiter » le processus de conception d'un jeu :

> *« Les habitudes de la programmation Orienté Objet m'amènent à exprimer instinctivement une « règle » comme une « propriété » d'un « élément de jeu ». Je pense que cette habitude est inappropriée : chaque « règle » devrait être représentée par un objet propre, et d'un point de vue conceptuel cette règle devrait même être une entité qui « contrôle » les « éléments de jeu » tout en pouvant les modifier. Mais notre représentation habituelle d'une « règle » ne correspond pas du tout à cela. A la place, ces dernières se trouvent dispatchées à divers endroits du code. Il est rarissime, dans le code gérant le « gameplay » d'un jeu, de trouver une « règle » qui soit intégralement implémentée à un seul endroit du code. En d'autres termes, il est rare de trouver une relation directe entre « une règle de jeu » et « un morceau de code ». […] Si nous pensons que notre code sera plus propre et plus facile à maintenir si son architecture correspond à celle de ce qu'il représente conceptuellement, alors nous devons changer notre manière de programmer le code gérant le gameplay du jeu. Nous devons migrer du modèle Orienté Objet vers un paradigme dans lequel les « éléments de jeu » n'intègrent pas les « règles » qui les concernent, car ces dernières seront implémentées à travers des « objets » indépendants. »*[105]

Le recours à l'approche « objets intelligents » n'est pas problématique dans le contexte industriel où l'équipe de création du jeu associe développeurs et Game Designers. A l'inverse, dans un contexte pédagogique impliquant un groupe d'étudiants ne possédant aucune notion de programmation informatique, cela apparaît comme une limite. A moins que le but recherché ne soit d'enseigner la programmation *« Orienté Objet »*, autant utiliser un outil dont la manière de modéliser la structure ludique est la plus proche possible de celle d'un Game Designer. Ainsi, nous choisissons d'opter pour l'approche « objets conteneur de données » pour la réalisation de notre outil.

---

[105] *"The point is that the habits of Object-Oriented programming lead me to instinctively express this rule as properties or behaviors of the nouns. I think this habit is inappropriate: each rule should be represented as an object in its own right; and conceptually the rule should in fact be an element that "controls" the nouns and can modify them. But our current typical representation of rules doesn't do this at all. Instead they find themselves poked and prodded into various places in code. It's rare, in gameplay code, to find a rule that is even concisely expressed in exactly one place in code - in other words, it's rare to find a 1-to-1 relationship between "a game rule" and "a chunk of code." […] If we believe that our code will be cleaner and more maintainable if it structure matches the structure of the conceptual objects that we're actually representing, then we should move gameplay code to a paradigm in which Nouns have no behavior of their own; and push to make Rules and Verbs first-order objects."*

Précisons toutefois que ce choix ne reflète que l'architecture logicielle qui sera utilisée pour construire l'outil technique. En effet, rien n'empêche d'imaginer par la suite une « interface auteur » qui permette de travailler séparément les *« règles »* pour chacun des *« éléments de jeu »*. Après tout, il existe autant de méthodes de conception que de concepteurs *(p.84)*. Mais même dans ce cas, l'approche « objets conteneurs de données » apparaît plus pertinente. En effet, il nous semble a priori plus facile d'implémenter une « interface auteur » donnant l'illusion que chaque *« élément »* possède ses propres « *règles* » sur une architecture logicielle où les *« éléments »* et les *« règles »* sont séparées, que de donner l'illusion d'une grande liste de *« règles »* indépendante à partir d'un amas de *« règles »* intégrées dans chacun des *« éléments de jeu »*. Si nous conservons donc la possibilité d'avoir plusieurs « interfaces auteur » différentes pour offrir à l'utilisateur le choix de son approche de conception *(p.257)*, l'architecture logicielle de notre outil sera bien basée sur le modèle « objets conteneurs de données ».

1.4 UN SEUL « EDITEUR VISUEL » POUR PROGRAMMER TOUTES LES REGLES DE JEU

En ce qui concerne le mode de création du *« Compute »* *(p.193)*, nous souhaitons nous affranchir de la nécessité d'apprendre un langage de programmation. Une des principales raisons de ce choix est l'accessibilité auprès d'un large public par la réduction du niveau de compétences techniques requis. En effet, nous avons pu constater lors de nos cours avec *The Games Factory 2* *(p.251)*, qui n'utilise pas de langage de programmation, que les étudiants participaient bien plus largement à la programmation des *« règles de jeu »* que lorsque nous leur proposions d'utiliser *Flash* et son langage de script propriétaire *(p.249)*. De plus, à la fin d'un de nos cours, une de nos étudiante nous a formulé la remarque suivante : *« c'est dommage que vous n'ayez pas précisé qu'il ne fallait pas savoir programmer sur le descriptif du cours. J'ai plusieurs copines qui n'ont pas voulu venir avec moi à ce cours car elles pensaient qu'il fallait savoir programmer et ce n'est pas vraiment leur truc. Mais avec The Games Factory 2 on n'a pas besoin de programmer pour créer un jeu, donc quand je leur ai raconté ce que nous avions fait cette semaine elles ont regretté de ne pas être venues. »*. Bien qu'anecdotique, cette remarque illustre bien qu'en réduisant le niveau de compétences techniques requis pour créer des jeux vidéo, cette activité peut s'adresser à un public plus large et plus varié. Pour la création des *« règles de jeu »*, le choix d'un *« éditeur visuel »* au lieu d'un langage de programmation nous semble donc primordial en regard du contexte d'utilisation pédagogique que nous visons.

Après avoir choisi l'approche *« éditeur visuel »*, un autre choix s'offre à nous : la portée de cet éditeur. En effet, s'il existe des outils comme *The Games Factory 2* qui proposent uniquement un *« éditeur visuel »*, y compris pour la création de règles complexes, d'autres logiciels comme *RPG Maker* proposent un *« éditeur visuel »* limité à la création des règles les plus simples. Pour la création de règles plus complexes, il faut passer par un langage de script propriétaire. Là encore, nous avons vu que cette approche est plutôt vécue comme une limite par nos étudiants *(p.251)*. Nous proposons donc d'utiliser uniquement un *« éditeur visuel »*, y compris pour la création des règles les plus complexes *(opérations sur plusieurs variables issues d'objets, accès de « bas niveau » aux composants du jeu…)*. Loin d'être anodin, ce choix conditionne le potentiel d'application de notre outil vers le côté « pédagogie » plutôt que « industrie ». En effet, pour les professionnels de l'industrie du Serious Game, le fait d'avoir un *« éditeur visuel »* suffisamment puissant pour être autonome s'avère moins pertinent que le recours à un *« éditeur visuel »* limité aux tâches simples, mais couplé à un langage de programmation pour les tâches complexes. Par exemple, la société ***Succubus*** possède un outil interne qui permet aux Game Designers de programmer rapidement les règles des Serious Games qu'ils réalisent[106]. Basé sur un *« éditeur visuel »* qui n'est pas sans

[106] Source : discussion avec ***Laurent Auneau*** lors de la conférence professionnelle *E.Virtuoses*, 31-05-11.

rappeler celui de *Thinking Worlds* *(p.214)*, cet outil ne permet pas aux concepteurs de créer des règles trop complexes. Par exemple, les *« blocs d'instructions »* proposées par l'outil seront *« lorsque la variable du personnage X est égale à Y »* ou *« Lancer le dialogue Z »*. Mais si le concepteur veut rajouter des *« blocs d'instructions »* qui ne sont pas inclus dans le logiciel, par exemple pour *« faire rebondir un personnage qui touche un mur »*, cela est impossible. En effet, la société estime que ce type de règles est trop complexe à implémenter pour un Game Designer, et qu'il s'agit là d'un travail pour lequel un programmeur sera plus efficace. Le Game Designer ira donc voir un programmeur pour qu'il réalise les nouveaux *« blocs d'instructions »*» qu'il aura imaginé. La société ***OKTAL*** procède de la même manière avec son logiciel *SCANeR* : *« l'éditeur visuel »* n'est là que pour les règles simples, l'invention de nouvelles règles complexes implique de passer par l'équipe de développement *(p.219)*. Même les chercheurs travaillant dans le domaine semblent partager cette vision. Par exemple, *Raptor* est un outil de prototypage rapide destiné aux Game Designers de l'industrie du jeu vidéo de divertissement (John David Smith, 2009; John David Smith & N. Graham, 2010). Il permet à un Game Designer de simuler en temps réel les règles de son jeu, en manipulant directement les divers *« éléments de jeu »* sur une table interactive alors qu'un joueur est train de tester le jeu. Tel un marionnettiste, le Game Designer peut ajouter ou supprimer des éléments à la volée, les déplacer, ou leur demander d'effectuer des actions telles que *« tirer »*, *« sauter »*… Malgré les nombreuses possibilités offertes au concepteur par cet outil, si ce dernier veut que ses *« éléments de jeu »* effectuent une action qui n'est pas présente dans le système, il est obligé de demander à un développeur de lui programmer un nouveau *« bloc d'action »*.

Ainsi, pour les applications industrielles, le fait de limiter la puissance de *« l'éditeur visuel »* au profit d'un langage de programmation semble être la solution la plus efficace. Et en effet, que ce soit pour des raisons d'optimisation de code ou même de rapidité de production, il semble que programmer des interactions complexes à bases de « blocs » de bas niveau tels ceux proposés par *Scratch* *(p.189)* ou *The Games Factory 2* demande bien plus de temps de travail à un Game Designer qu'à un programmeur rédigeant du code. Mais cela implique un pré-requis inévitable : qu'il y ait au moins une personne maîtrisant un langage de programmation informatique dans le groupe de création du Serious Game. Si cela est toujours le cas dans l'industrie, c'est beaucoup plus rare pour le secteur de l'éducation. Que ce soit pour nos cours ou pour ceux dispensés par d'autres personnes *(p.237)*, les apprenants ont rarement un programmeur expérimenté à leur disposition. Même si cela n'est pas l'approche la plus pertinente en terme de productivité, le fait de proposer un *« éditeur visuel »* suffisamment puissant pour permettre au concepteur d'inventer toutes les règles de jeu qu'il souhaite semble indispensable dans un contexte pédagogique *(à l'exception évidente des cours où le but recherché est l'apprentissage de la programmation)*. De plus, cette approche semble présenter un autre avantage fondamental pour la création, comme nous allons le voir dans le point suivant.

1.5 UN OUTIL « GENERALISTE » MAIS « SPECIALISABLE » AU BESOIN

La distinction la plus fondamentale que nous avons opérée lors de l'analyse des usines à jeux est la séparation des usines à jeux spécialisées dans un seul genre vidéoludique de celles qui permettent de créer tout genre de jeu vidéo, voire d'en inventer de nouveaux *(p.192)*. Sur le terrain, les usines à jeux « spécialisées » permettent d'aller bien plus vite car les jeux qu'elles permettent de créer possèdent automatiquement certaines règles de jeu « standard » au genre, à l'inverse des usines à jeux « généralistes » pour lesquelles toutes les règles doivent être construites explicitement. Si l'approche « généraliste » s'avère la plus intéressante en terme de créativité, voire même d'apprentissage *(p.251)*, l'approche « spécialisée » s'impose quand il s'agit de s'adresser à un large public en réduisant la complexité technique de la création

vidéoludique. La solution idéale serait alors d'essayer de trouver un compromis entre ces deux approches. Mais est-ce seulement possible ?

Après avoir inventé le célèbre *Pinball Construction Set* *(p.168)*, ***Bill Budge*** chercha à réaliser un outil de création qui s'affranchisse du carcan d'un genre vidéoludique unique. Son projet *Construction Set Construction Set* visait à inventer un outil toujours basé sur un éditeur visuel pour rester simple d'accès, mais qui permette néanmoins de créer tout type de jeu vidéo. Face à la complexité de la tâche avec la puissance des ordinateurs des années 80, ***Budge*** abandonna rapidement son projet (Hague, 1997). Pourtant, ce concept semble encore l'inspirer de nos jours, alors qu'il occupe le poste de *« programmeur principal des outils de développement »* au sein de la société ***Sony Computer Entertainement***. Son projet a maintenant évolué pour devenir une collection de composants que les programmeurs peuvent utiliser pour créer facilement des *« éditeurs visuels »* à destination des Game Designers de la société[107]. Ce pionnier des usines à jeux semble donc encore être en mesure de nous « montrer la voie ». En effet, si un outil « généraliste » est trop complexe tandis qu'un outil « spécialisé » est trop limité, pourquoi ne pas imaginer un outil « généraliste » que l'on pourrait « spécialiser » au besoin pour le rendre plus accessible ?

Nous avons déjà évoqué dans le point précédent le fait que les industriels du Serious Game utilisaient des outils spécialisés : *SCANeR* est dédié à la simulation automobile, l'outil de ***Succubus*** est destiné à la création de jeux d'aventure graphique et *Raptor* limité aux jeux d'action en vue subjective *(p.260)*. Ces outils sont particulièrement simples d'accès et rapides à utiliser. Mais si les concepteurs de jeu veulent créer des règles qui n'ont pas été « anticipées » lors de la création de l'outil, ils doivent demander aux programmeurs de leur créer de nouveaux « blocs » *(ou apprendre la programmation s'ils veulent le faire par eux-mêmes)*. En d'autres termes, l'approche « spécialisée » pour les outils reposant sur un *« éditeur visuel »* est plus facile à utiliser jusqu'à ce que le concepteur ait besoin de « blocs » qui ne sont pas prévus par l'outil. Dans ce cas, pour atténuer la complexité de la création de nouveau « blocs », ne pourrait-on pas imaginer les créer en utilisant un *« éditeur visuel »* d'outil « généraliste » plutôt qu'un langage de programmation ?

C'est justement l'approche mise en avant par *Virtools* à travers sa fonctionnalité de *« blocs de comportement »* *(p.182)*. L'utilisateur peut ainsi créer de nouveaux « blocs » en définissant ses variables d'entrées et de sortie, puis peut utiliser les *« blocs de base »* du logiciel pour créer la logique interne au *« bloc de comportement »*. En conséquence, les *« blocs de base »* de cet outil sont de relativement bas niveau : *opération simples sur des variables, gestion directe des périphériques d'entrée et de sortie…* Utilisé directement pour construire un jeu, un tel outil est plus complexe d'approche qu'une usine « spécialisée » : ici, le concepteur doit construire ses règles en assemblant de nombreux éléments de base. Mais qu'est-ce qui empêcherait par exemple un enseignant de construire de nombreux *« blocs de comportement »* de haut niveau en utilisant ces *« blocs de base »* de bas niveau, pour ensuite les proposer à ses élèves afin de leur faciliter la création d'un Serious Game ? Ainsi, pour la majorité des cas où les « blocs de haut niveau » sont suffisants, les élèves ont à leur disposition un outil aussi simple d'emploi qu'une usine à jeux « spécialisée ». Mais contrairement à ces dernières, si un des élèves à besoin de modifier les « blocs de haut niveau » à sa disposition, voire d'en créer de nouveaux, il n'aura pas cette fois besoin d'apprendre un langage de programmation. Il pourra le faire en utilisant l'unique *« éditeur visuel »* du produit, qu'il a déjà utilisé pour construire son jeu. La création d'un nouveau *« bloc de comportement »* demandera certes du temps à l'élève, mais moins que si ce dernier avait du utiliser un langage de programmation qu'il ne connaît pas. Et cela évite également la situation, particulièrement frustrante pour les élèves, de ne pas pouvoir réaliser leurs idées car

[107] Source : échanges personnels par mail avec ***Bill Budge***, 19-05-10.

« le logiciel ne propose pas le bloc d'instruction adéquat ». En d'autres termes, en permettant à un outil « généraliste », basé sur un *« éditeur visuel »*, de laisser l'utilisateur créer ses propres « blocs », il est possible d'envisager un outil qui conserve la puissance d'un outil « généraliste » tout en s'appropriant la plus grande facilité d'accès des outils « spécialisés ». Au-delà de *Virtools*, nous pouvons également retrouver cette approche dans le module *Build Your Own Block* de *Scratch* *(p.189)*. Face au succès rencontré par cette variante créée par un amateur, l'équipe à l'origine de *Scratch* a décidée de l'ajouter comme fonctionnalité de base dans la future version *2.0* du logiciel *(p.210)*. Enfin, la nature « open-source » de *Scratch* et de son module additionnel fait qu'ils ont été intégrés au récent *StencylWorks* ***(Stencyl, 2011)***. Cette usine à jeux « généraliste » permet à ses utilisateurs de créer des *« comportements »* complexes, tels que *« mouvement d'un personnage selon les codes d'un jeu de plateforme »*, à partir de *« blocs de base »* de bas niveau *(comparaison et modification des objets, lecture de l'interface entrante…)*. Ces *« comportements »* peuvent être associés à plusieurs *« variables »* permettant de les configurer, par exemple pour changer la vitesse de déplacement ou la hauteur de saut du personnage auquel aura été appliqué le comportement *« mouvement d'un personnage selon les codes d'un jeu de plateforme »*. De plus, la nature « Jeu 2.0 » *(p.200)* de *StencylWorks* permet à ses utilisateurs de partager facilement les *« comportements »* qu'ils créent. Les utilisateurs de ce logiciel bénéficient donc d'une simplicité d'accès comparable à une usine à jeux « spécialisée » tout en conservant la puissance d'un outil « généraliste ».

Au final, nous pouvons voir dans ces derniers exemples une concrétisant du projet *Construction Set Construction Set* originel de ***Bill Budge***. En permettant à l'utilisateur de créer ses propres « blocs d'instruction », il est en quelque sorte possible de « spécialiser » au besoin une usine à jeux « généraliste ». Face à l'énorme potentiel de cette approche, nous souhaitons bien évidemment l'implémenter au sein de notre outil.

1.6 L'INTEGRATION D'OUTILS THEORIQUES

Nous avons déjà pu constater la difficulté de transmettre les outils théoriques imaginés par les Game Designers *(p.59)* à un public qui n'est pas spécialisé dans la conception vidéoludique, qu'il s'agisse de professionnels « experts d'un domaine » *(p.87)* ou d'étudiants réalisant leur premier jeu vidéo *(p.248)*. Aussi utiles et pertinents que soient ces outils, ils sont souvent jugés « trop complexes » par un public novice en Game Design. Une manière de faire bénéficier ce public de l'apport des outils théoriques semble être de les intégrer au sein d'un outil technique. Il existe des logiciels basés sur un modèle théorique de conception vidéoludique, à l'image de *Machinations* *(p.70)* ou encore de *Ludocore*. Ce dernier outil permet de modéliser la structure d'un jeu en utilisant le langage logique *« Event Calculus »* (Adam M Smith, Nelson, & Mateas, 2010). Aussi intéressants soient-ils, ces deux outils s'adressent plutôt aux professionnels du Game Design. En effet, au-delà de la complexité du langage de modélisation qu'ils utilisent, ils ne permettent pas à proprement parler de « créer » un jeu vidéo. Ils sont plutôt destinés à évaluer automatiquement la cohérence d'une structure de jeu, de manière à y déceler des problèmes potentiels avant même d'en réaliser le premier prototype. Pour s'adresser à un public novice, l'intégration logicielle d'un outil théorique doit plutôt être un moyen de « guider » l'utilisateur lors de la réalisation de son projet de Serious Game. Nous avons déjà rencontré quelques exemples pensés dans cette optique au sein des travaux de recherche pourtant sur les outils de conception de Serious Games. Par exemple, le langage visuel *WEEV* intégré à *<e-Adventure>* *(p.222)*, la collection de *« widgets »* imaginés par *Marfisi-Schottman & al.* *(p.91)* ou tout simplement *« l'écran à idées »* de *Adventure Author* *(p.225)*. En regard des premiers résultats observés sur ces trois outils et du potentiel de leur démarche, l'intégration d'un outil théorique de conception de Serious Game au sein de notre outil technique semble donc pertinente. La question reste cependant ouverte quant à la nature de l'outil théorique à intégrer. En effet, en plus des nombreux outils théoriques étudiés pour le Serious Game *(p.86)* et le jeu vidéo *(p.59)*, nous avons également développé notre

propre réflexion sur le sujet tout au long de cette thèse *(p.150)*. De plus amples réflexions sur les outils théoriques semblent donc nécessaires avant d'en intégrer un, voire même plusieurs, au sein de notre outil technique.

1.7 UN OUTIL PENSE POUR LE PROTOTYPAGE

Dans la plupart des méthodologies de conception de Serious Game que nous avons étudiées *(p.103)*, la notion de prototypage est particulièrement importante. Afin de pouvoir évaluer la pertinence de leurs concepts de jeu auprès du public ciblé, nos apprenants ont également besoin de réaliser de nombreux prototypes. Le fait de faciliter et d'accélérer la réalisation de jeux vidéo est déjà en soi un moyen de favoriser le prototypage. D'après ***Fullerton*** (2008), même les professionnels de l'industrie du jeu vidéo ont souvent recours aux usines à jeux pour tester rapidement leurs idées sans mobiliser des ressources en développement informatique. Un des principaux besoins lors de l'évaluation est alors de pouvoir corriger rapidement le prototype suite aux retours des joueurs, afin de pouvoir le re-évaluer. ***Fullerton*** cite en exemple le prototype d'un jeu de stratégie qui a été conçu pour pouvoir être corrigé durant la partie. Les concepteurs du jeu peuvent ainsi ajuster les règles du jeu directement pendant la phase de test, gagnant alors un temps précieux. En plus d'essayer de faciliter la création vidéoludique, notre outil devra donc dans l'idéal permettre de modifier le jeu créé « à la volée ». A notre connaissance, seuls deux logiciels proposent une telle particularité parmi les 400 outils que nous avons analysés : *Stagecast Creator* *(p.180)* et *Scratch* *(p.189)*. Cette fonctionnalité, qui semble pourtant intéressante, est donc pour l'instant loin d'être très répandue.

1.8 FACILITER LE « SUIVI DU JOUEUR » A DES FINS D'EVALUATION

Autre fonctionnalité encore peu répandue mais qui semble particulièrement importante pour la conception de Serious Games, le suivi du joueur. Nous avons déjà discuté de l'intérêt des techniques de « suivi du joueur » pour évaluer l'impact d'un Serious Game de type « intrinsèque » *(p.145)*. Parmi les trois approches de « suivi de joueur » que nous avons évoquées, il nous semble intéressant d'essayer d'intégrer au moins l'approche utilisant des « outils de suivi générique » à notre outil. Ainsi, les étudiants pourraient facilement analyser les données enregistrées pour leur jeu grâce aux nombreux modes de visualisation proposés par ces outils génériques *(p.147)*. L'intégration à un *LMS* semble également pertinente pour élargir la base d'utilisateurs potentiels de notre outil, bien que cette voie limite sévèrement la qualité des données enregistrables *(p.145)*.

1.9 UNE APPROCHE « JEU 2.0 »

Un des retours récurrent de nos étudiants lors de nos cours est le souhait d'avoir accès à une « base commune » d'animations, sons et autres moteurs de jeu pour les aider dans leur travail *(p.253)*. L'approche « Jeu 2.0 » *(p.200)* nous semble la meilleure des réponses à ce besoin. En effet, en intégrant notre outil à une plateforme de partage, tous les jeux déjà créés, que ce soit en terme de programmation, d'animation, de sons, de moteur de jeu… deviendraient alors des ressources pour les étudiants des promotions suivantes. Certes, nous proposons déjà une base de données des jeux réalisés par les promotions précédentes au format open-source *(p.249)*. Mais les étudiants travaillant sur des outils techniques différents au fil des années *(p.248)*, cela limite parfois l'intérêt de cette initiative. Alors qu'avec une plateforme « Jeu 2.0 », tous les jeux sont au même format car créés avec le même outil, facilitant ainsi l'échange de ressources entre étudiants de promotions différentes.

Toujours dans l'optique d'une approche « Jeu 2.0 », notre outil devra être réalisé avec une technologie facilitant sa diffusion sans installation par le biais d'Internet. Cela représente un véritable avantage pour l'utilisation pédagogique d'un outil technique. En effet, les réseaux informatiques des établissements interdisent généralement l'installation de logiciels sur leurs machines. Mais avec un outil de création utilisable directement dans un navigateur Internet, il n'y a pas d'installation, et donc point de blocage de la part du réseau de l'établissement. Si plusieurs technologies intéressantes existent dans le domaine, lors de la rédaction de cette thèse il en existe une qui domine largement : *Flash* *(p.186)*. La plupart des « Jeu 2.0 » destinés aux ordinateurs que nous avons étudiés s'appuient sur cette technologie *(p.202)*. Et les quelques exemples qui n'y sont pas encore ont tendance à migrer vers elle pour faciliter leur diffusion, à l'image de la version « 2.0 » de *Scratch* qui abandonne *Java* pour *Flash*. Si ces deux technologies sont maintenant comparables en terme de puissance et possèdent un langage de programmation très proche, la plus grande diffusion de *Flash* lui confère pour l'instant un avantage sur ses concurrents. Nous réaliserons donc vraisemblablement notre outil à partir de cette technologie.

1.10 PERMETTRE LE TRAVAIL COLLABORATIF

Nous avons également pu remarquer lors de nos cours que les étudiants, réunis en groupes, essayaient au maximum de travailler en parallèle *(p.248)*. Nous tenterons donc de permettre au maximum le travail de plusieurs personnes en simultané sur un même projet de Serious Game.

1.11 L'INTEGRATION D'UN « TUTORIAL » D'UTILISATION SOUS FORME DE JEU

Enfin, nous avons constaté qu'apprendre aux joueurs comment jouer à un Serious Game était parfois aussi important que le jeu en lui-même *(p.147)*. Cette observation s'applique également aux outils techniques, particulièrement quand ils se destinent à des novices en création vidéoludique. La plupart des outils restent très « classiques » à ce niveau, en proposant simplement un manuel et des fichiers d'exemples. Certes, il existe sur Internet des tutoriaux et autres ressources créés par les communautés d'utilisateurs de ces outils techniques *(p.170)*. Mais il existe également quelques outils qui essaient d'innover en proposant un tutorial d'utilisation de l'outil sous forme de jeu vidéo. Par exemple, comme nous le précisait ***Llanas***, avant de créer des niveaux avec l'éditeur de *LittleBigPlanet*, il est plus que recommandé de terminer le jeu qui l'accompagne *(p.206)*. Cela permet de découvrir tous les éléments qui pourront être utilisés pour construire des niveaux. *Gamestar Mechanic* propose un principe similaire, un des objectifs du jeu intégré à cet outil étant de sensibiliser les joueurs aux problématiques de Game Design *(p.210)*. *Stagecast Creator* intègre également un tutorial interactif sous forme de jeu, profitant pour cela de sa fonctionnalité de « modification du jeu à la volée » *(p.264)*. Dans un autre registre, signalons que le constructeur ***AMD*** a financé la création d'un site Internet communautaire proposant des « défis », qui sont en fait des tutoriaux permettant d'apprendre à utiliser le logiciel *Game Maker* *(p.179)*. Baptisé *Activate!*[108], ce site *« Web 2.0 »* permet aux utilisateurs de partager les jeux qu'ils créent une fois qu'ils ont réussi tous les « défis » du site. Cette initiative originale mise donc sur la dimension communautaire pour motiver les collégiens américains à créer des jeux vidéo, dans l'espoir de les inciter à s'orienter vers des études relatives aux sciences et à l'ingénierie. Bien que les modalités exactes restent à définir, voire à inventer, l'idée d'intégrer un tutorial sous forme ludique pour faciliter l'apprentissage de notre outil semble donc tout à fait pertinente.

[108] http://www.activategames.org/

Mentionnons pour terminer une idée intéressante bien que moins fondamentale. Comme nous l'exprimait ***Llanas*** dans son retour d'expérience sur l'utilisation de *LittleBigPlanet* *(p.206)*, le caractère « ludique » de la manipulation d'un outil de création peut également être un facteur motivant pour des élèves. Si pour les étudiants de nos cours cet aspect ne semble pas primordial, il semble des plus importants pour s'adresser à un public plus jeune. Ainsi, *Scratch* *(p.189)*, qui s'adresse avant tout aux enfants, possède une interface très ludique. La création de règles se fait par l'association de *« blocs »* manipulés directement à la souris, donnant un aspect « puzzle » très amusant à la création des règles. Même si elle ne sera pas au coeur de nos priorités, le fait de proposer une interface « amusante » semble donc un moyen d'élargir l'âge du public « novice » qui pourra potentiellement utiliser notre outil dans un contexte d'apprentissage. Loin d'être anecdotique, cette question renvoie aux très nombreuses considérations spécifiques au design des interfaces de logiciels (Crawford, 2002).

2 TENTATIVES DE REALISATION

Fort des nombreux choix et idées centralisés dans ce cahier des charges, nous allons maintenant essayer de réaliser un outil technique qui réponde à ces spécifications. La nature « générale » des idées du cahier des charges impliquera vraisemblablement des choix supplémentaires dictés par des contraintes techniques lors de la réalisation de l'outil. Le processus de recherche au cœur de la création d'un tel logiciel dépasse donc le cadre de cette thèse. Avançant par versions itératives, la réalisation d'un outil de création vidéoludique est une tâche de très longue haleine. A titre d'exemple, l'équipe de chercheurs à l'origine de *<e-Adventure>* *(p.222)* aura mis plus de six ans pour réaliser un logiciel aussi complet[109]. Nous présentons donc dans cette thèse les deux premiers pas qui, nous l'espérons, permettrons un jour de réaliser un logiciel répondant à l'ensemble des points évoqués dans le cahier des charges.

La première initiative présentée ici concerne la réalisation du logiciel *Gam.B.A.S.*, dont le déroulement est antérieur à la rédaction du cahier des charges. Cette première expérimentation logicielle fut d'ailleurs une des principales motivations pour sa rédaction. Son successeur, le projet *LudoForge*, a donc démarré sur des bases plus solides. Bien que loin d'être terminée à la date d'écriture de cette thèse, cette seconde expérimentation logicielle a permis la réalisation d'un « moteur d'interprétation » fonctionnel *(p.257)*. Cette première version de *LudoForge* fut alors éprouvée à travers la réalisation de jeux vidéo et Serious Games *(p.289)*. Ce chapitre détaille les principales caractéristiques de ces deux expérimentations logicielles. Précisons qu'afin d'envisager une application « Jeu 2.0 » *(p.264)*, ces deux outils techniques ont été développés avec la technologie *Flash* et son langage de programmation *« Orienté Objet »*. Les diagrammes de classes des différentes versions de ces logiciels sont disponibles dans les annexes de cette thèse *(p.307)*.

[109] Source : discussion avec ***Pablo Moreno-Ger*** lors de la conférence *1st GALA Alignement School*, 20-06-11.

Le projet *Gam.B.A.S* s'est déroulé entre 2006 et 2008. Il visait à la création d'un outil technique s'appuyant sur l'outil théorique des *« briques de GamePlay »*. Trois prototypes distincts de cet outil technique ont été réalisés, avant l'abandon définitif du projet.

2.1.1 LES « BRIQUES DE GAMEPLAY »

Les *« briques de GamePlay »* sont un outil théorique, que nous avons inventé avec ***Julian Alvarez***, pour permettre une lecture simplifiée des règles de jeu vidéo. Par exemple, si nous analysons les « règles du jeu » de *Pac-Man* ***(Namco, 1980)*** et de *Space Invaders* ***(Taito, 1978)***, nous identifions les règles suivantes: *« Si Pac-man touche un fantôme, alors détruire Pac-man »* et *« Si vaisseau du joueur touche un tir ennemi, alors détruire vaisseau du joueur »*. Nous observons ici une grande similarité entre ces deux règles, et pouvons considérer qu'elles sont construites sur la base suivante *« Si l'élément joueur touche un élément hostile, alors le jeu renvoie un jugement négatif au joueur »*. Afin de simplifier les analyses ultérieures, nous avons choisi d'associer un « verbe » à cette « règle de base », ici le verbe *« Eviter »*. Selon un processus similaire, nous avons pu identifier dix « règles de base » en analysant un corpus de 588 jeux vidéo. Une fois associée à un verbe, ces « règles de base » sont baptisées « briques ». Ces dix « briques » se répartissent en deux catégories : les briques *« Game »*, qui sont associées à des règles définissant des objectifs à atteindre, et les briques *« Play »* qui sont associées à des règles définissant des moyens et contraintes pour atteindre ces objectifs :

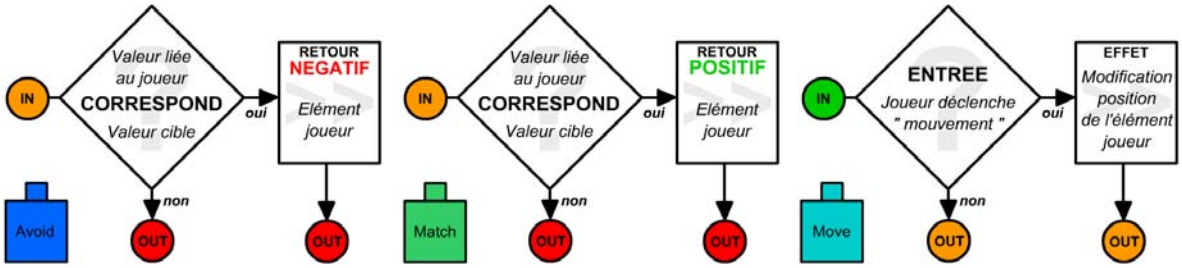
- **Briques « Game » :** *Eviter, Atteindre, Détruire.*
- **Briques « Play » :** *Créer, Gérer, Déplacer, Choisir, Tirer, Ecrire, Aléatoire.*

94. Les dix *« briques de GamePlay »*

Ces dix briques ont été originellement conçues comme la base d'un système de classification des jeux vidéo centré sur le *« gameplay »*[110]. La genèse de ces briques ainsi que leur application pour le système de classification *V.E.Ga.S.* sont détaillées dans nos travaux de Master 2 Recherche (Djaouti, 2007) ainsi que dans la thèse de ***Julian Alvarez*** (2007). Ces briques sont aujourd'hui utilisées comme critère d'analyse du « gameplay » dans notre système de classification des Serious Games *G/P/S* *(p.28)*.

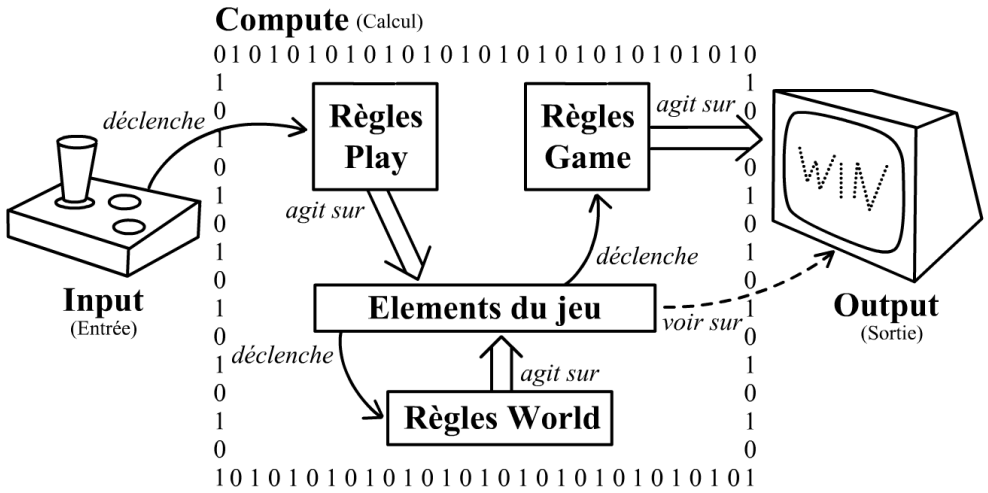
Au-delà de sa dimension « classification et analyse », cet outil théorique est également applicable à la conception de jeu vidéo. Nous nous appuyons sur la vision d'un jeu vidéo en tant que « système à état variable » composé « d'éléments » et de « règles » *(p.156)*. L'idée de formaliser des « règles de base » abstraites permet alors d'établir une sorte de « grammaire » des « règles ». Cette approche rejoint le concept de *« ludemes »* introduit par ***Koster*** *(p.69)*, certaines des *« Game Design Patterns »* *(p.79)*, ou plus récemment les travaux sur une *« sémiotique des mécanismes de jeu »* (Vick, McDaniel, & Jacobs, 2010). Concrètement, chacune de nos *« briques de GamePlay »* est définie sous forme schématique de manière à proposer un « canevas » *(« template »)* permettant de construire des règles de jeu :

[110] Terme anglophone ne possédant pas de traduction française directe. Historiquement, ce mot est dérivé de l'expression *« How the game plays ? »*, qui était le titre des fiches d'instructions se trouvant sur les premières bornes d'arcade. Aujourd'hui, ce terme renvoie généralement au principe de jeu ou à des notions connexes.



95. Exemples de définition schématique des *« briques de GamePlay »* : *Eviter, Atteindre et Déplacer*

Les *« briques Play »* sont basées sur un canevas permettant de créer des « règles » modifiant l'état des « éléments » en réponse à l'interface entrante. Les *« briques Game »* sont destinées à formuler un retour négatif ou positif au joueur à partir de l'état courant des « éléments » du jeu. Pour permettre la création d'un jeu vidéo fonctionnel, cette typologie de « règles » est complété d'une troisième catégorie : les « règles » qui agissent sur les « éléments » en réponse à l'état des « éléments ». Il s'agit de « règles » s'appliquant indépendamment de l'action du joueur, par exemple des règles permettant de simuler la gravité terrestre. Nous baptisons ces règles les *« règles World »*, ou « règles de monde ». Le schéma ci-dessous, qui reprend trois des composantes du *modèle ISICO* *(p.159)*, illustre la relation entre les différents types de « règles » et les « éléments » d'un jeu vidéo :



96. Typologie des règles en action dans la partie *« Compute »*

Les *« briques de GamePlay »* ne permettent de créer que des règles de type *« Game »* ou *« Play »*. Cela introduit donc une limite dans leur utilisation à des fins de conception. Ainsi, les *« briques de GamePlay »* peuvent uniquement aider le concepteur pour la création de certains types de règles, la gestion des « éléments » et des règles *« World »* sortant de leur cadre d'application. Tout en étant bien conscients de cette limite, nous avons tenté de réaliser un outil technique de Game Design qui s'appuie sur les *« briques de GamePlay »* pour faciliter le travail du concepteur : *Gam.B.A.S.*

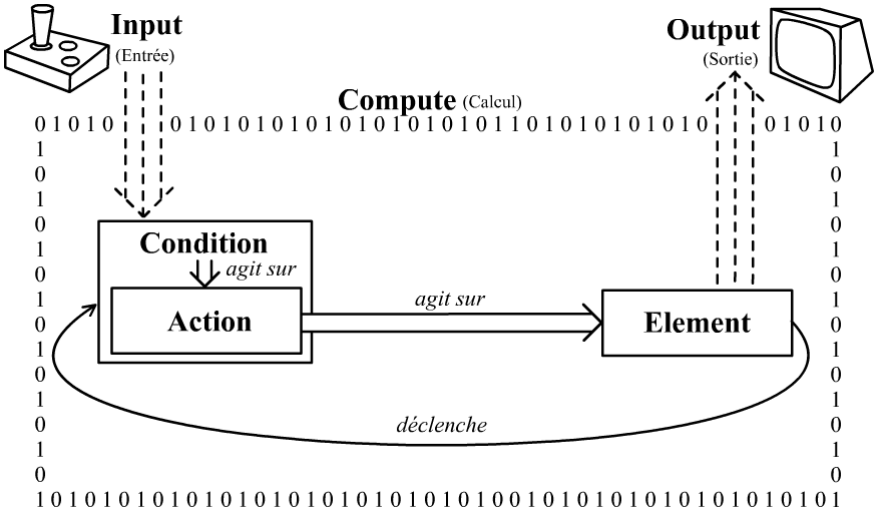
2.1.2 GAM.B.A.S. - PROTOTYPES #1 ET #2 « VERSION 1.0 »

La conception et la réalisation du premier prototype de *Gam.B.A.S.* est détaillée dans nos travaux de Master 2 Recherche (Djaouti, 2007).

2.1.2.1 Architecture générale

Pour réaliser ce prototype nous avons tout d'abord du imaginer une architecture logicielle permettant de modéliser les « éléments » et les « règles » d'un jeu vidéo. Nous nous sommes pour cela appuyé sur le modèle *« Orienté Objet »* *(p.257)* et avons choisi de représenter les « éléments » et les « règles » par des classes d'objets différentes *(p.258)*. La classe d'objet

« Elément » contient diverses « propriétés » permettant d'enregistrer « l'état courant » de l'élément *(position X et Y, taille, couleur…)*. Les règles sont quant à elles modélisées à partir de deux classes d'objets complémentaires : la classe *« Condition »* et la classe *« Action »*. Nous rejoignons là le paradigme de la quasi-totalité des outils techniques proposant un *« éditeur visuel »* pour la création des « règles » *(p.177)*. Selon ce paradigme, une « règle de jeu » est vue comme l'association de conditions *(SI Pac-man touche un fantôme…)* et d'actions *(…ALORS détruire Pac-man)*. La classe mère *« Condition »* possède deux propriétés : une liste d'instances de la classe *« Action »* et une liste d'instances de la classe *« Elément »*. La seule méthode de la classe mère *« Condition »* s'appelle *« appliquer() »*. Elle a pour effet de vérifier si la nature de la condition *(collision, comparaison de valeurs…)* est « vraie » pour ses instances de la classe *« Elément »*. Si oui, elle exécute la méthode *« appliquer() »* de chacune de ses instances de la classe *« Action »*. De son côté, la classe mère *« Action »* ne possède qu'une seule propriété : la référence à une instance de la classe *« Elément »*. Elle possède également une méthode abstraite *« appliquer() »* qui a pour effet de modifier les « propriétés » de l'instance *« Elément »* qu'elle référence *(changement des propriétés de position X et Y, de couleur, de taille, destruction de l'instance…)*.



97. Interaction des différentes classes mères de *Gam.B.A.S. 1.0* à travers le *modèle ISICO*

A partir de ces classes mères, de nombreuses conditions et actions ont pu être créées. Pour créer une condition, par exemple *« Collision »*, qui permet de détecter la collision entre deux *« Elément »*, il suffit de créer une nouvelle classe *« Collision »* qui hérite de la classe mère *« Condition »*. Pour déterminer le statut « vrai » ou « faux » d'une condition, nous nous appuyons sur un modèle de programmation événementielle. Pour toutes les instances de *« Elément »*, chaque cycle de la boucle principale du programme effectuera quelques tests standards : *tests de collisions entre tous les objets, enregistrement des actions de l'utilisateur sur l'interface entrante (clavier, souris)…* Selon le résultat de ces tests, la boucle principale génèrera alors un événement précis pour chaque type de test, en précisant les instances ou paramètres associés à l'événement : *paires d'instances en collision, code de la touche du clavier appuyée, notification du démarrage de la partie…* Lorsqu'une instance d'une classe héritant de *« Condition »* est créée, elle s'ajoute automatiquement comme écouteur de l'évènement qui la concerne. Par exemple, chaque fois qu'un évènement de collision sera envoyé, chaque instance de la condition *« Collision »* vérifiera que les deux instances *« Elément »* signalées comme étant en collision par la boucle principale correspondent aux deux instances *« Elément »* définies dans ses paramètres internes. Si les instances *« Elément »* de l'évènement sont les mêmes que celles de l'instance de la condition, alors cette dernière appliquera la liste d'actions qu'elle contient. Pour créer une action, il suffit de définir une nouvelle classe héritant de la classe mère *« Action »*. Par exemple pour créer l'action *« Déplacer »*, il suffit de définir la méthode *« appliquer() »*, qui est abstraite dans la classe

mère. Chaque action possède ainsi un code spécifique dans la méthode *« appliquer() »*. Pour *« Déplacer »*, cette méthode modifie tout simplement les « propriétés » correspondant à la position X et Y de l'instance de la classe *« Elément »* à laquelle s'applique l'action. La démarche est similaire pour la création « d'éléments » : il suffit de créer une nouvelle classe d'objet héritant de la classe mère *« Elément »*, en définissant les propriétés souhaitées pour ledit élément.

2.1.2.2 Modélisation des « briques de GamePlay »

Nous avons ainsi défini plusieurs classes de conditions et d'actions, ainsi qu'une classe « d'élément » : un carré possédant des propriétés pour sa position en X et en Y, sa taille et sa couleur. A partir de ces différentes classes d'objet, il nous a été possible de réaliser plusieurs petits jeux vidéo simples. Par exemple un jeu dans lequel le joueur contrôle un « carré vert » qui doit aller toucher des « carrés violets » pour les détruire, tout en évitant de toucher des « carrés bleus ». Pour cela, nous avons tout d'abord créé des instances de la classe *« Carré »* qui hérite de *« Elément »*. Ensuite, nous avons généré des instances de classes héritant de *« Condition »* auxquelles nous avons assigné une liste d'instances de classes héritant de *« Action »*. Par exemple, nous avons ainsi créé une « règle » composée de la condition *« AppuiClavier »*, paramétrée pour se déclencher en réponse à la touche « flèche gauche », et de l'action *« Déplacer »*, paramétrée pour soustraire la valeur « 3 » à la propriété « position X » de l'instance de la classe *« Carré »* de couleur verte. Ainsi, quand le joueur appuiera sur la touche « flèche gauche » de son clavier, le carré vert se déplacera de trois pixels vers la gauche. Suivant un principe similaire, nous avons défini toutes les « règles » et « éléments » permettant de réaliser notre petit jeu vidéo à base de carrés de couleur.

Nous avons alors utilisé ce petit jeu comme une base permettant d'implémenter le concept des *« briques de GamePlay »*. Pour cela nous avons défini une classe mère *« Brique »*, à partir de laquelle hériterons les différentes classes représentant chacune de nos dix briques. Le constructeur de chaque classe héritant de *« Brique »* permet de définir les « règles » associées à cette brique. En effet, nous avons évoqué tout à l'heure le fait que les *« briques de GamePlay »* sont des « canevas » *(« templates »)* permettant de générer des « règles de jeu », comme le reflète d'ailleurs la nature « générique » des schémas les définissant *(p.267)*. Mais pour implémenter cet outil théorique dans un jeu vidéo, nous avons besoin de les « spécifier » plus précisément. Par exemple, la brique *« BriqueDéplacer »* est simplement définie comme *« SI action du joueur sur l'interface entrante ALORS modifier position spatiale d'un élément »*. Lors de l'implémentation de ce « canevas » dans un jeu, son concepteur doit définir explicitement la nature de *« l'action du joueur sur l'interface entrante »* ainsi que les valeurs et cibles exactes de la *« modification spatiale d'un élément »*. Nous retrouvons là une problématique similaire à celle du travail de ***Levieux*** sur la mesure de la difficulté dans un jeu vidéo : afin de mesurer des capacités génériques telles que « tirer », il est obligé de définir pour chaque jeu quelles variables à utiliser comme référence *(p.145)*. Mais si nos *« briques de GamePlay »* doivent être « spécifiées » lors de la conception de chaque jeu, comment proposer un outil permettant au concepteur de les utiliser pour créer plusieurs jeux différents ?

Pour tenter de répondre à cette question, chaque brique est définie comme une classe d'objet dont une des propriétés contient une liste d'instances de classes héritant de *« Condition »*, tandis que les autres propriétés sont des paramètres destinés à « configurer » ces règles. Pour revenir sur l'exemple de la brique *« BriqueDéplacer »*, son constructeur crée quatre instances de la condition *« AppuiClavier »*, configurées pour détecter les quatre touches fléchées du clavier. Il créé ensuite quatre instances de l'action *« Déplacer »*, qui sont attribuées à chacune des quatre conditions *« AppuiClavier »*. Elles sont également configurées pour ajouter ou soustraire une valeur « Z », en accord avec la direction de la touche fléchée de la condition à laquelle l'action est associée. La valeur « Z » ainsi que la référence à une instance de *« Elément »*, qui sont nécessaires pour que l'action *« Déplacer »* fonctionne, ne sont par

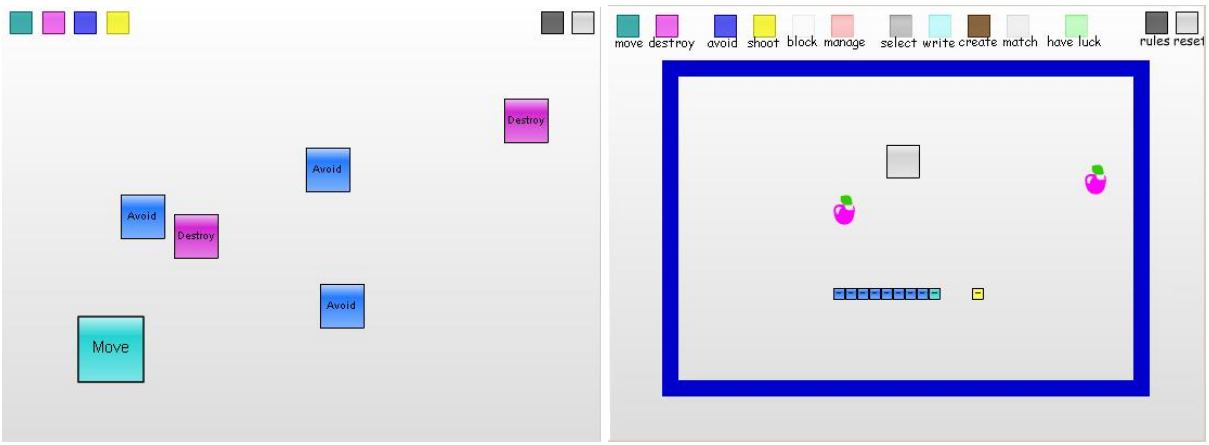
contre pas définies directement par le constructeur. A la place, la valeur « Z » et l'*« Elément »* ciblé par les quatre règles sont attribués par référence aux paramètres de la brique. Ainsi, lorsque que le concepteur crée une instance de *« BriqueDéplacer »*, il passe au constructeur deux paramètres : une instance de *« Elément »* et une valeur numérique. En choisissant uniquement ces deux paramètres, le concepteur du jeu crée donc automatiquement quatre « règles » de jeu qui permettent de déplacer l'élément qu'il souhaite en utilisant les flèches du clavier. Cette approche nous semblant intéressante pour tenter de simplifier la création vidéoludique, nous avons implémenté quatre briques selon ce principe : *« Déplacer »*, *« Atteindre »*, *« Eviter »* et *« Tirer »*. Nous avons alors essayé de recréer le petit jeu du « carré vert » qui doit toucher des « carrés violets » en évitant des « carrés bleus » avec ces briques. Au lieu d'avoir à définir une par une les nombreuses règles de ce jeu, le fait d'utiliser les quatre *« briques de GamePlay »* implémentés dans ce premier prototype simplifient effectivement le travail du concepteur. Elles lui permettent de travailler à un niveau légèrement plus « haut ». Ainsi, pour réaliser son jeu vidéo, concepteur doit uniquement préciser au logiciel *« je veux que cet élément puisse se déplacer »*, au lieu d'avoir à préciser *« quand j'appuie sur la touche gauche la position X de cet élément est diminuée de 5 pixels »* et ainsi de suite pour toutes les directions de mouvement souhaitées. Si à travers ces briques le concepteur perd le contrôle direct sur la modélisation des règles de son jeu, il gagne en temps et en simplicité de réalisation.

2.1.2.3 Implémentation de la totalité des « briques de GamePlay »

Face aux débuts encourageants que représente le premier prototype de *Gam.B.A.S.*, nous avons décidé de continuer l'expérience afin de réaliser une version plus complète qui implémente toutes les *« briques de GamePlay »*. Comme cela est détaillé dans la thèse de ***Julian Alvarez*** (2007), ce travail a été réalisé dans le cadre d'un « projet tutoré » proposé à deux étudiants en Master 1 d'informatique. Afin d'impliquer ces deux étudiants motivés par nos recherches, nous leur avons tout d'abord exposé le principe théorique des *« briques de Gameplay »*. Nous leur avons ensuite proposé de nous aider à créer un outil technique les implémentant. Pour cela, nous leur avons fourni le code source du premier prototype de *Gam.B.A.S.* comme base de travail.

Cependant, nous avons déjà évoqué que, pour être utilisé à des fins de conception, les *« briques de GamePlay »* doivent être « spécifiées » de manière précise. Prenons par exemple la brique *« Déplacer »*. D'un point de vue théorique, cette brique est très pertinente pour représenter le fait que le joueur peut déplacer un « élément » par le biais de l'interface entrante. Mais lors de la création d'un jeu vidéo, son concepteur doit préciser la manière dont ledit élément va se « déplacer ». Pourra t'il se déplacer dans quatre directions différentes suite à l'appui de quatre boutons différents comme dans *Pac-man* ? Ou faudra t'il de cliquer sur un endroit de l'écran pour que l'élément s'y dirige automatiquement comme dans *Warcraft II* ***(Blizzard Entertainment, 1995)*** ? Si conceptuellement nous avons bien là deux cas de « déplacement », dans la pratique les « règles » qui permettent à ces « éléments » de se déplacer sont très différentes. Dans le cas de *Pac-man*, il suffit d'incrémenter ou de décrémenter de manière continue les propriétés correspondants aux positions X et Y de l'élément. Dans le cas de *Warcraft II*, il faut enregistrer les coordonnées X et Y auxquelles l'utilisateur a cliqué, puis incrémenter ou décrémenter de manière séparée les propriétés correspondants aux positions X et Y de l'élément jusqu'à ce qu'il atteigne les coordonnées du clic. Comment proposer alors une brique *« Déplacer »* qui permette de créer des mouvements aussi différents ? Et encore, ces deux exemples sont limités à des jeux en 2D. Comment incorporer la dimension de mouvement supplémentaire introduite par les jeux en 3D ? Il semble illusoire d'imaginer une brique *« Déplacer »* suffisamment paramétrable pour permettre de créer tout les types de « déplacement » existants dans les jeux vidéo. Et quand bien même cela serait possible, comment alors permettre au concepteur d'inventer de nouvelles manières pour le joueur de « déplacer » un élément ?

Nous avons déjà amené une première réponse à cette question à travers le premier prototype de *Gam.B.A.S.* Bien que configurable, sa classe d'objet *« BriqueDéplacer »* ne modélise qu'un type particulier de déplacement. A ce stade de nos travaux, il semble donc pertinent d'imaginer non pas une brique *« Déplacer »* qui serait universelle, mais une multitude de briques *« Déplacer »* correspondant à des mouvements différents pour créer autant de jeux vidéos différents. Pour la réalisation du second prototype de *Gam.B.A.S.*, nous avons donné un objectif très simple aux étudiants : reproduire le célèbre *jeu du serpent* tout en implémentant toutes les *« briques de GamePlay »*. Les étudiants ont donc repris le premier prototype et ont programmé de nouvelles classes héritant de *« Action »* telles que *« SuivreElement »* ou *« CiblerElement »*. En utilisant les classes héritant de *« Condition »* et *« Action »*, ils ont pu définir de nouvelles classes héritant de *« Brique »* dont les constructeurs créent automatiquement plusieurs « règles » à partir des paramètres définit par le constructeur. Ils ont finalement poursuivi le travail entamé sur le premier prototype en implémentant la totalité des briques, de manière à faciliter la création de variantes du *jeu du serpent*. Pour créer un jeu avec cet outil, l'utilisateur doit pour l'instant saisir des lignes de code pour créer les instances des différentes *« Brique »*. Il s'agit donc à proprement parler plus d'un « framework » que d'une véritable usine à jeu basée sur un *« éditeur visuel »*. Mais les deux premiers prototypes de *Gam.B.A.S.* disposent néanmoins de « boutons » permettant d'activer ou désactiver automatiquement toutes les instances d'une classe héritant de *« Brique »*. Par exemple, en appuyant sur un interrupteur, il est possible d'activer ou désactiver la possibilité pour le « carré vert » ou le « serpent » de tirer des projectiles. Le fait d'appuyer sur ce bouton active où désactive simplement la prise en compte, par la boucle principale du programme, des « règles » contenues dans chacune des instances de *« BriqueTirer »*. A défaut d'avoir un *« éditeur visuel »* permettant de créer de nouveaux jeux vidéo, les deux premiers prototypes de *Gam.B.A.S.* proposent donc un « jeu vidéo modifiable » accessible à tous les joueurs.



98. *Gam.B.A.S.* : prototype #1 et prototype #2

2.1.2.4 Bilan de cette expérimentation

Au final, la conception et la réalisation de ces deux prototypes ont permis de réfléchir à une modélisation de la structure d'un jeu à partir d'une approche *« Orienté Objet »* *(p.257)*, tout en respectant au maximum la manière de penser d'un Game Designer *(p.258)*. Bien qu'ils ne proposent pas *« d'éditeur visuel »* *(p.260)*, ces deux logiciels implémentent un outil théorique destiné à guider la création vidéoludique *(p.263)* en la forme de nos *« briques de GamePlay »*. Afin d'implémenter cet outil théorique, nous avons cependant été contraints d'opter pour une approche de type « usine à jeux spécialisée » *(p.261)* afin de répondre à la nécessité de « spécifier » les briques pour un genre de jeu donné.

Au-delà de ces quelques manquements à notre « cahier des charges », l'architecture même de *Gam.B.A.S.* souffre de nombreuses lacunes. Par exemple, le fait que les « règles » soient

modélisée par un objet *« Condition »* incorporant plusieurs instances de *« Action »* implique que chaque « règle » du jeu ne peut pas posséder plusieurs conditions. Avec ce modèle, il est donc impossible de déclencher le mouvement de personnage suite à l'appui d'une touche sur le clavier tout en vérifiant si l'élément est en collision avec un autre élément *(par exemple pour créer un objet que le joueur peut déplacer tant qu'il ne sort pas d'une zone précise)*. De plus, le fait que la boucle principale teste de manière exhaustive tous les évènements potentiels, même si aucune « règle » existante dans le jeu n'utilisera certains de ces évènements, est aberrant d'un point de vue optimisation. Le programme est ainsi conduit à évaluer, à chaque cycle, des collisions inutiles entre éléments ou de tester les états de périphériques d'entrée inutilisés. Certes, nous sommes pour l'instant dans une optique d'expérimentation logicielle et non d'optimisation, mais ce choix d'architecture introduit potentiellement une sévère limite quant à la fluidité de l'outil technique résultant. Dans un autre registre, les règles ne s'appliquent pour l'instant qu'à des instances précises des classes héritant de *« Elément »*. Si un concepteur veut donc appliquer la brique *« Déplacer »* à tous les *« Carré »* du jeu, il devra créer et appliquer autant d'instances de la brique *« Déplacer »* qu'il existe d'instances de la brique *« Carré »*. De plus, si une instance de *« Carré »* est créée dynamiquement, le concepteur ne possède aucun moyen de lui appliquer une brique, étant donné que *Gam.B.A.S.* ne permet pas de modifier les règles « à la volée » *(p.264)*. Si ce choix peut tout à fait se justifier d'un point de vue technique *(un programme « propre » ne devant appliquer des règles que sur des instances existantes)*, il est contradictoire en regard de notre volonté de proposer un logiciel permettant de modéliser un jeu vidéo d'une manière se rapprochant de la manière de penser d'un Game Designer *(p.258)*. En effet, ce dernier aurait plutôt tendance à imaginer une règle de type *« tous les éléments Carré peuvent se déplacer »*, plutôt que de spécifier une multitude de règles *« l'élément carré #X peut se déplacer »*. Aussi intéressante soit-elle, cette première expérimentation de réalisation d'un outil technique nous amène donc à continuer nos travaux dans une direction différente.

2.1.3 GAM.B.A.S. - PROTOTYPE #3 « VERSION 2.0 »

La conception et la réalisation du troisième prototype du projet *Gam.B.A.S.* marque le début du travail de recherche de cette thèse. Bien qu'intéressants, les deux premiers prototypes réalisés durant nos travaux de Master 2 Recherche ont montré de nombreuses limites dues à l'architecture choisie. Afin de tenter de dépasser ces limites, nous avons effectué un choix radical : tout reprendre à zéro. Nous avons défini une nouvelle architecture pour le logiciel, et l'avons entièrement reprogrammé. Destiné à permettre l'exploration de nouvelles approches, ce logiciel reste cependant basée sur le modèle *« Orienté Objet »* *(p.257)*. De plus, il vise toujours à représenter les « éléments » et les « règles » par des classes d'objets différentes *(p.258)*, même si la manière de les modéliser a évoluée. Les diagrammes de classes illustrent les nombreuses évolutions d'architecture entre les deux versions de *Gam.B.A.S.* *(p.315)*.

2.1.3.1 Gestion des « éléments »

Du côté des « éléments », la nouvelle classe *« Elément »* permet dorénavant de « simuler » différentes classes différentes à la volée. Chaque « élément » d'un jeu est défini par une liste de propriétés dont seul le nombre, les noms et les types varient. Par exemple, un *« vaisseau spatial »* sera représenté par trois variables numériques : sa position X, sa position Y, et son nombre de points de vie restants. Les *« lasers »* que tire ce vaisseau seront quant a eux représentés par quatre valeurs numériques : position X, position Y, direction de déplacement et vitesse de déplacement. Dans les premiers prototypes, il fallait créer deux classes d'objets différentes pour représenter ces deux « éléments de jeu ». Dorénavant, il est possible de les représenter tous deux directement avec la nouvelle classe *« Elément »*. Cette classe permet de créer de nouvelles propriétés dynamiquement pour chaque élément qu'elle représente. Ainsi le constructeur de la classe *« Elément »* permet d'attribuer un « nom de classe virtuel » désignant la catégorie de l'élément, tel que *« vaisseau »* ou *« laser »*. Il permet également de donner une liste de noms de propriétés en précisant leur type ainsi que leur valeur de départ.

Le « moteur d'expression », que nous détaillerons ultérieurement *(p.275)*, permet aux instances d'*« Elément »* de posséder des propriétés pouvant stocker quatre types de variables : les valeurs numériques *(float)*, les chaînes de caractères alphanumériques *(string)*, la référence à un autre objet de la classe *« Elément »*, ainsi qu'une liste de plusieurs valeurs d'un des trois types précédents *(array)*. Cette nouvelle version permet donc enfin à *Gam.B.A.S.* de gérer les références entre « éléments » et les variables scalaires *(p.257)*.

2.1.3.2 Gestion des « règles »

Pour les « règles », nous avons créé une classe *« Règle »* permettant de représenter proprement les règles de jeu sous forme d'objet distinct *(p.258)*. Cette classe possède deux propriétés : une liste d'instances héritant de la classe mère *« Condition »*, ainsi qu'une autre liste destinée à recevoir des instances héritant de la classe mère *« Action »*. L'existence d'une classe séparée pour les règles autorise dorénavant la création de « règles » possédant plusieurs conditions.

Comme dans la version précédente, les classes mères *« Condition »* et *« Action »* permettent de créer de nombreuses conditions et actions en écrivant des classes qui en héritent. Mais le fait que les règles puissent posséder plusieurs conditions a cependant modifié leur fonctionnement général, car nous nous appuyons toujours sur un modèle de programmation évènementielle. Les *« Condition »* écoutent directement les évènements émis par la boucle principale du programme. Imaginons par exemple une règle comme *« SI l'utilisateur clique ET que deux éléments sont en collision, ALORS détruire les deux éléments »*. Cette règle possède deux conditions simultanées : le fait que l'utilisateur clique, et le fait que deux « éléments » soient en collision. Or, dans notre modèle évènementiel, les évènements correspondant à ces deux conditions ne pourront que très rarement être émis en même temps. L'évènement de collision entre deux objets est émis lorsque deux objets qui n'étaient pas en collision entrent en contact, tandis que l'évènement correspondant au clic ne sera émis qu'une seule fois lorsque l'utilisateur clique. Le seul moyen pour que les deux conditions de cette règle soient « vraies » serait donc qu'un joueur clique sur la souris lors du cycle durant lequel les deux « éléments » entrent en collision. A raison de 25 cycles à la seconde, il est quasi-impossible qu'un joueur puisse cliquer à ce moment exact. En l'état, cette condition ne sera donc jamais « vraie ». Une solution évidente à ce problème serait de modifier notre modèle de gestion des évènements. Par exemple, en modifiant l'évènement de collision pour qu'il soit envoyé à chaque cycle tant que les objets sont en collision. Bien qu'efficace pour ce cas précis, cette solution est plutôt coûteuse en terme de performances : elle génère l'envoi de nombreux évènements inutiles. Nous avons donc opté pour une autre approche. La première des *« Condition »* qui reçoit un évènement de la part de la boucle principale appelle la méthode *« appliquer() »* de la *« Règle »* qui la contient. Cette *« Condition »* passe également une référence à elle-même à cette méthode pour s'identifier en tant que déclencheur. La *« Règle »* va alors évaluer une à une la validité de toutes les *« Condition »* qu'elle contient, à l'exception du déclencheur qui est déjà « vrai » puisque étant à l'origine de l'appel. Pour revenir à notre exemple, si le joueur clique après que deux *« Eléments »* soient en collision, la *« Condition »* détectant le clic recevra un évènement et déclenchera l'application de la *« Règle »*. Cette dernière évaluera alors si les deux *« Eléments »* sont en collision à ce moment précis. Si oui, les deux conditions sont validées, et la règle appliquera donc la liste des *« Action »* qu'elle possède.

Le choix d'utiliser une condition comme déclencheur possède également un autre avantage. Prenons cette fois-ci l'exemple d'une règle *« SI des éléments sont en collision ET que la position Y de ces éléments est supérieure à 10, ALORS détruire ces éléments »*. La condition déclenchant cette *« Règle »* sera obligatoirement la condition *« Collision »*, la condition *« ComparerPropriétés »* n'étant pas associée au système d'évènement pour des raisons évidente d'optimisation *(l'envoi d'un évènement à chaque changement de chaque variable*

d'un logiciel étant plus que coûteux en terme de ressources). Supposons que la condition *« Collision »* reçoive un évènement lui notifiant la collision des *« Elément »* A et B, sachant que le jeu contient pour l'instant une centaine d'instances de *« Elément »*. Une fois que *« Collision »* sait que A et B sont en collision, comment peut-elle communiquer à *« ComparerPropriétés »* le fait qu'elle doivent tester la position Y de ces deux *« Elément »* ? En effet, si *« ComparerPropriétés »* décide de tester la position Y de tous les *« Elément »* du jeu, cela revient à effecteur 98 tests inutiles. Il serait plus efficace de permettre à *« Collision »* de préciser quels sont les *« Elément »* qu'elle vise pour que *« ComparerPropriétés »* ne teste que ceux-là. C'est là que notre système de *« Condition »* faisant office de déclencheur en appelant la méthode *« appliquer() »* devient pertinent. Lorsque une *« Condition »* appelle cette méthode de la *« Règle »* qui la contient, elle fournit également comme paramètre la liste des *« Elément »* la vérifiant. Ainsi *« Collision »* fournit la liste des deux *« Elément »* qu'elle a détecté comme étant en collision à sa *« Règle »*. Cette dernière passera alors en paramètre cette liste de *« Elément »* à toutes les autres *« Condition »* qu'elle contient. Mais ce système ne s'arrête pas là. Supposons maintenant que *« Collision »* ait en fait détecté non pas deux, mais trois *« Elément »* en contact. Elle donne alors la liste de ces trois *« Elément »* à sa *« Règle »*, qui la transmet à son tour à *« ComparerPropriétés »*. Lorsque *« ComparerPropriétés »* teste leur position Y, elle constate que seulement deux des trois *« Elément »* ont une position Y supérieure à 10. En l'état, la *« Règle »* n'est donc validée que partiellement. Comment *« ComparerPropriétés »* peut-elle alors communiquer le résultat de ses tests ? De la même manière que *« Collision »*. Lorsque une *« Règle »* teste ses *« Condition »*, elle leur fournit une liste de *« Elément »*, mais attend également une telle liste en retour. Si la *« Condition »* à l'origine du déclenchement de la *« Règle »* fournit une première liste de *« Elément »*, toutes les autres conditions de cette même *« Règle »* agissent donc comme un « filtre » permettant d'éliminer progressivement les *« Elément »* qui ne vérifient pas toutes les *« Condition »* de la *« Règle »*. Si, après toutes ces vérifications, la liste de *« Elément »* n'est pas vide, alors la *« Règle »* exécute la fonction *« appliquer() »* de chacune de ses *« Action »*, en leur passant comme paramètre la liste des *« Elément »*. La *« Règle »* sera donc bien appliquée uniquement aux *« Elément »* qui la vérifient.

Ce système, certes complexe à expliquer, permet d'obtenir une gestion des « règles » bien plus puissante que celle de la première version de *Gam.B.A.S.*. Il se conjugue avec un système plus élaboré de « sélection des éléments » détaillé ci-après *(p.275)*. Rappelons si besoin est que lorsque nous mentionnons les *« Condition »* et *« Action »* dans notre explication, nous faisons référence aux nombreuses classes telles que *« Collision »* ou *« ComparerPropriétés »* qui héritent de la classe mère *« Condition »* et à *« SupprimerElément »* ou *« ModifierPropriétés »* qui héritent de la classe mère *« Action »*.

2.1.3.3 Développement d'un « moteur d'évaluation d'expression »

Une des limites des premiers prototypes de *Gam.B.A.S.* était le fait que les « règles » ne pouvaient cibler que des instances précises de *« Elément »* *(p.272)*. Pour appliquer une règle à trois instances *d'« Elément »*, il fallait donc créer trois règles différentes. Mais le véritable problème lié à ce type d'architecture survient lorsqu'un nouvel *« Element »* est créé par les règles du jeu. Le concepteur n'a alors aucun moyen d'appliquer globalement des règles à toutes les instances de *« Elément »* d'un même type, que ces instances soient générées au début du jeu ou durant la partie. Supposons par exemple qu'un concepteur imagine un jeu dans lequel le joueur dirige une « balle bleue » qui doit éviter de toucher des « balles rouges » qui rebondissent à l'écran. Toutes les cinq secondes, une nouvelle « balle rouge » apparaît pour compliquer la tâche du joueur. Pour créer un tel jeu, un concepteur imaginera vraisemblablement les « règles » suivantes : *« SI un élément « balle bleue » entre en collision avec une « balle rouge » ALORS détruire « balle bleue » »* et *« SI compte à rebours est égal à cinq secondes ALORS créer nouvelle « balle rouge » »*. Comme nous souhaitons proposer un outil technique qui se rapproche du mode de pensée d'un Game Designer *(p.258)*, nous

devons proposer une architecture permettant de formaliser les règles d'une manière proche de celle-ci. Pour cela, nous devons mettre en place un système permettant aux *« Règle »* de sélectionner facilement plusieurs instances de *« Elément »* par rapport à leur type. Nous avons alors imaginé une classe *« CiblageElement »*, qui permet de saisir une expression textuelle, par exemple « Ballebleue ». Cette expression textuelle sera alors transformée en une liste d'instances d'*« Elément »* correspondants à chaque évaluation de la règle. Concrètement, lorsque une *« Condition »* ou une *« Action »* est évaluée par sa *« Règle »* et qu'un de ses paramètres contient une instance de *« CiblageElement »*, la méthode *« évaluer() »* de *« CiblageElement »* est appelée. En utilisant comme référence l'expression textuelle saisie par le concepteur, par exemple « Ballebleue », *« CiblageElement »* cherche alors au sein de toutes les instances de *« Elément »* existantes dans le jeu, et construit une liste de références aux instances dont le type est égal à « Ballebleue ». Cette liste est ensuite passée à la *« Condition »* ou *« Action »* qui est en train d'être appliquée. Cette dernière l'utilisera pour évaluer ses critères de validité ou pour appliquer ses effets à tous les éléments de la liste. Ainsi, le concepteur du jeu n'aura qu'à formaliser une seule *« Règle »* de la forme *« SI un élément « balle bleue » entre en collision avec un élément « balle rouge » ALORS détruire « balle bleue » »*. Cette règle s'appliquera automatiquement à toutes les instances de *« Elément »* de type « balle rouge » et « balle bleue » du jeu. Et ce, même si des instances de *« Elément »* sont dynamiquement créées ou supprimées durant la partie.

Toutes les *« Condition »* et *« Action »* qui doivent cibler des *« Elément »* attendent donc une instance de la classe *« CiblageElement »* comme paramètre. Mais cette problématique d'évaluation des « règles » ne s'arrête pas au seul ciblage des « éléments ». Prenons par exemple une *« Règle »* dont une des *« Condition »* de la classe *« ComparerPropriétés »*. Cette *« Condition »* doit vérifier si un nombre aléatoire compris entre un et cinq est égal à deux afin que la *« Règle »* soit appliquée. De prime abord, nous pourrions penser qu'il suffit de donner comme paramètre à *« ComparerPropriétés »* d'un côté la valeur « 2 » et de l'autre la valeur « aléatoire(1,5) ». Cependant, cette solution comporte un énorme problème. Avec ce principe, la valeur « aléatoire(1,5) » ne sera évaluée qu'une seule fois lors de la création de l'instance de *« ComparerPropriétés »*, par son constructeur. Si, lors de cette unique évaluation, la valeur aléatoire retourne « 3 », alors la *« Règle »* ne sera jamais appliquée. Pour obtenir un comportement correspondant aux attentes du concepteur, il faut que la valeur aléatoire soit recalculée à chaque fois que la condition *« ComparerPropriétés »* est évaluée. Pour cela, nous avons créé une classe *« Nombre »* destinée à renvoyer une valeur numérique à chaque évaluation d'une *« Règle »*, d'une manière analogue à *« CiblageElement »*. La classe *« Nombre »* attend une expression textuelle comme paramètre lors de sa construction, et utilisera ensuite cette expression pour renvoyer une valeur numérique lors de l'appel de sa méthode *« évaluer() »*.

Afin d'homogénéiser la gestion des paramètres des *« Condition »* et *« Action »*, nous avons finalement décidé que tous les paramètres seraient saisis en utilisant de telles classes d'objets conteneurs. Nous avons donc créé une classe mère *« Expression »*, de laquelle hérite entre autres *« CiblageElement »*, *« Nombre »* ainsi que *« NombreAléatoire »*. Reprenons à présent notre exemple de la condition *« SI valeur aléatoire entre un et cinq est égale à deux »*. Pour créer cette condition avec une instance de *« ComparerPropriétés »*, le concepteur passera comme paramètres au constructeur une instance de *« NombreAléatoire »* avec le paramètre « 1-5 » et une instance de *« Nombre »* avec le paramètre « 2 ». A chaque fois que cette condition sera évaluée par sa *« Règle »*, l'instance de *« Nombre »* renverra toujours une valeur de 2. De son côté, *« NombreAléatoire »* renverra bien une valeur aléatoire comprise entre 1 et 5 qui sera différente à chaque fois. Selon ce principe, nous avons créé de nombreuses classes héritant de *« Expression »*, telle que *« Liste »*, *« Texte »*, *« PropriétéElément »*... Toutes ces classes attendent une expression textuelle comme paramètre de construction. Elle utiliseront cette expression textuelle lors de l'appel de leur méthode *« évaluer »* pour renvoyer

respectivement un tableau de valeur, une chaîne de caractères alphanumériques, ou une liste de valeurs extraites d'une propriété particulière d'un type d'*« Elément »* spécifié.

Face à cette collection de classes renvoyant des valeurs de manière dynamique, il faut mettre en place un système permettant de s'assurer de la cohérence entre les différents « types » de valeurs manipulées par le logiciel. Pour rappel, nous avons défini que les propriétés des *« Elément »* peuvent être de quatre types *(p.273)* : un nombre *(float)*, un texte *(string)*, une référence à un *« Elément »*, ou une liste *(array)*. Chaque classe héritant de *« Expression »* doit alors explicitement spécifier le type de valeur que renverra sa méthode *« évaluer() »*. Par exemple, *« Nombre »* et *« NombreAléatoire »* renvoient toujours une valeur de type « nombre », alors que *« Liste »* et *« CiblageElement »* renvoient une valeur de type « liste ». La prise en compte de ce paramètre permet à la condition *« ComparerPropriétés »* et à l'action *« ModifierPropriétés »* de s'assurer que les paires de valeurs qu'elles comparent ou modifient soient bien du même type. Afin de renforcer ce système, nous avons même externalisé la gestion des comparaisons et opérations d'éléments d'un même type. En effet, l'action *« ModifierPropriétés »* n'est pas la seule action à avoir besoin de par exemple additionner deux « nombres » ou deux « textes ». Pour ne pas multiplier le code de manipulation des valeurs, celui-ce se trouve uniquement à l'intérieur des classes *« Nombre »*, *« Texte »* et *« Liste »* héritant de *« Expression »*. Nous avons alors créé une interface *« TypeDeDonnée »* possédant deux méthodes abstraites : *« comparerValeurs() »* et *« modifierValeurs() »*. Ces méthodes ont ensuite été implémentées pour chacune des trois classes, avec un code spécifique à la comparaison ou aux opérations entre valeurs numériques, chaînes de textes et tableaux de données. Cela nous permet de centraliser le code de gestion d'un type de données particulier au sein d'une seule classe d'objets, facilitant ainsi la maintenance ultérieure du logiciel.

L'introduction de classes héritant de *« Expression »* et leur utilisation systématique pour passer des paramètres aux différentes *« Condition »* et *« Action »* composant les *« Règle »* est la base de ce que nous appelons le « moteur d'évaluation d'expression ». Bien que complexe à expliquer, ce moteur permet aux concepteurs de modéliser relativement simplement des « règles de jeu » respectant la forme dans laquelle ils les ont imaginées. Reposant sur des règles au résultat entièrement dynamique, ce moteur permet également d'envisager la modification « à la volée » d'un jeu créé avec notre outil. Les valeurs des paramètres étant évaluées à chaque cycle de calcul à partir des paramètres saisis par le concepteur, rien n'empêche la modification des paramètres entre deux cycles de calcul pour changer les règles du jeu en pleine partie. Comme évoqué dans le « cahier des charges », cela représente un avantage pour l'utilisation d'un outil à des fins de prototypage *(p.264)*.

2.1.3.4 Modélisation des autres composantes du modèle ISICO

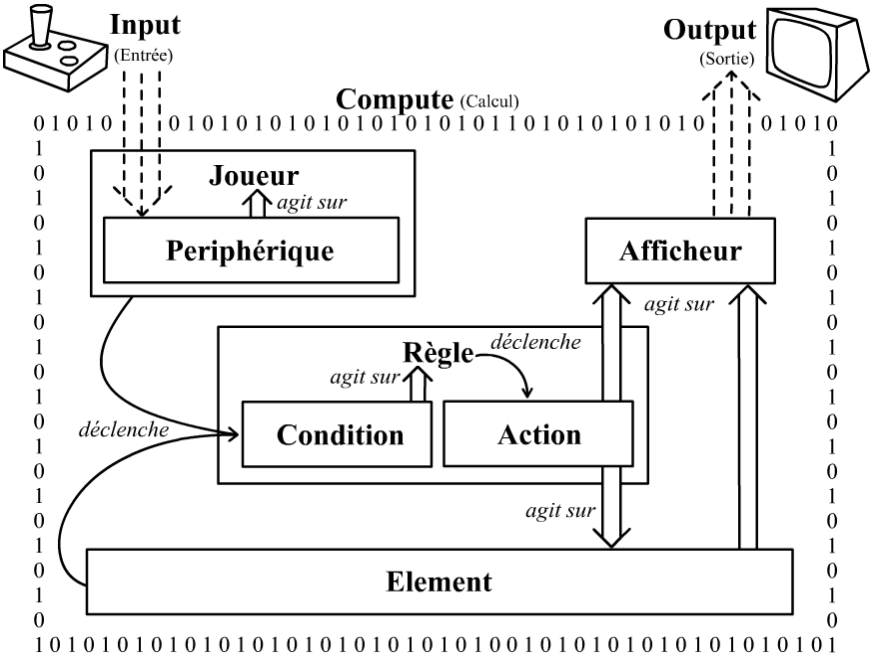
Avec cette nouvelle version du logiciel, nous avons également souhaité modéliser plus finement les composantes *« Input »* et *« Output »* du *modèle ISICO (p.159)*.

Abordons tout d'abord le côté *« Output »*, autrement dit la gestion de l'interface sortante. Dans les premiers prototypes, l'affichage était intimement lié aux *« Elément »* : ces derniers étaient dynamiquement associés à une représentation graphique sur laquelle il était possible de faire directement des tests de collisions *(il s'agit d'une des spécificités de la technologie Flash (p.264) pour la gestion des éléments graphiques)*. Pour cette nouvelle version de *Gam.B.A.S.*, nous avons décidé de proposer une architecture séparant explicitement les composantes *« Input »*, *« Compute »* et *« Output »* du *modèle ISICO*. Les classes *« Elément »* et *« Règle »*, ainsi que les classes héritant de *« Condition »* et *« Action »*, modélisent uniquement la composante *« Compute »*. Pour gérer la représentation graphique des *« Elément »*, et donc de la composante *« Output »*, nous avons introduit la classe mère *« Afficheur »*. De cette classe mère héritent les classes des différents moteurs de rendu

pouvant être intégrés au logiciel. Bien que nous nous limitions dans un premier temps à la création de jeux en 2D, nous anticipons ici la possibilité de connecter de manière transparente plusieurs moteurs de rendu graphique différents, par exemple 2D et 3D, à notre système de gestions des *« Elément »* et *« Règle »* de jeu. La classe *« Elément »* ne possède donc plus aucune information liée à la représentation de l'élément sur l'interface sortante. Ces dernières sont maintenant concentrées au sein de la classe *« Rendu2D »* héritant de *« Afficheur »*. Pour cela, *« Rendu2D »* possède une liste de *« Sprite »* représentant visuellement chacun des *« Elément »* du jeu. Lors de la création d'un nouvel *« Elément »* à travers l'action *« CréerElement »*, cette dernière envoie un évènement à toutes les instances de *« Afficheur »* et ses héritières pour leur notifier la création d'un *« Elément »*, et transmet toutes ses propriétés. *« Rendu2D »* va alors créer une nouvelle instance de *« Sprite »* et l'afficher à l'écran en tenant compte des propriétés de l'instance de *« Elément »* qu'elle représente. A chaque cycle, la boucle principale exécute alors la méthode *« actualiser() »* de *« Rendu2D »*. Cette méthode va mettre à jour l'apparence de chaque instance de *« Sprite »* en regard des propriétés de leur *« Elément »* respectif. Bien que les composantes *« Compute »* et *« Output »* soient maintenant séparées, elles gardent la possibilité de communiquer entre elles. Par exemple, la condition *« Collision »* n'est plus en mesure d'effectuer des tests de collisions car elle ne gère que des *« Elément »* sans représentation graphique. Pour tester une collision entre des *« Elément »*, cette condition envoie donc la liste des *« Elément »* qu'elle doit vérifier aux *« Afficheur »* associés au jeu *(p.280)*. L'*« Afficheur »* définit comme référence pour les collisions va alors tester les collisions entre les différents *« Sprite »* qu'il possède pour la liste des *« Elément »* que lui a envoyée la condition *« Collision »*. Il retourne une liste contenant uniquement les *« Elément »* qui sont en collision, de manière à ce que la *« Condition »* puisse continuer son travail au sein du *« Compute »* *(p.274)*. Un autre besoin de communication entre *« Compute »* et *« Output »* est lié à la gestion directe de l'interface sortante par certaines règles. Si les propriétés des *« Elément »* sont généralement suffisantes pour gérer leur représentation à travers des *« Sprite »*, il peut arriver qu'un concepteur souhaite parfois modifier l'interface sortante sans forcément que cela corresponde à la propriété d'un *« Elément »*. Par exemple, supposons qu'un concepteur de jeux souhaite faire clignoter la représentation graphique d'un *« Elément »* donné. Pour cela, il créera une règle comprenant l'action *« ModifierOutput »* qu'il configurera pour cibler les *« Elément »* de son choix et définira la modification comme « faire clignoter ». Cette action envoie un évènement à *« Rendu2D »*, qui va sélectionner les *« Sprite »* correspondants aux *« Elément »* demandés, et les fera dorénavant clignoter. La gestion des animations des *« Sprite »* passe également par ce système de communication entre *« Compute »* et *« Output »* par *« Action »* interposées.

A l'image de l'interface sortante, la gestion de la composante *« Input »* est également dissociée des deux autres. Elle repose sur la classe mère *« Joueur »*, qui modélise une représentation virtuelle des joueurs. De cette classe mère héritent plusieurs classes de joueurs telles que *« JoueurHumain »* et *« JoueurIntelligenceArtificielle »*. Comme son nom l'indique, *« JoueurHumain »* permet de gérer les joueurs humains et l'interface entrante qu'ils utilisent pour jouer. Pour cela, nous avons créé une autre classe mère : *« Périphérique »*. De cette classe mère héritent des classes représentant les différentes actions possibles sur les périphériques d'entrées, tels que *« AppuiToucheClavier »* ou *« ClicSouris »*. Une instance de *« JoueurHumain »* enregistre dans ses propriétés une liste d'instances de classes héritant de *« Périphérique »*. Chaque instance de *« Périphérique »* est associée à une chaîne alphanumérique représentant un « message ». Par exemple, une instance de *« AppuiToucheClavier »* configurée sur la touche « barre d'espacement » sera associée au message « tirer ». Lorsque qu'un joueur appuiera sur cette touche du clavier, l'instance de *« JoueurHumain »* communiquera à la boucle principale le message « tirer ». La boucle principale génèrera alors un évènement signalant à toutes les conditions de type *« ActionSurInterfaceEntrante »* que le message « tirer » à été émis. Toutes les conditions paramétrées pour réagir à ce message appliqueront donc la *« Règle »* qui les contient *(p.274)*.

Une subtilité supplémentaire s'ajoute à ce système. Chaque instance des classes héritant de *« Joueur »* possède également une propriété « cible », qui correspond à une expression de ciblage retournant une liste d'*« Elément »* *(p.275)*. Cela permet de limiter les actions des *« Joueur »* à certains éléments. La manière de rédiger les règles dans cette version de *Gam.B.A.S.* permet par exemple d'écrire *« SI un joueur déclenche l'action « tirer » pour les éléments de type « vaisseau spatial », ALORS créer élément « tir laser » et le déplacer selon un vecteur de mouvement (0,-10) »*. Cette règle ne spécifie pas quel joueur exécute l'action « tirer », ouvrant ainsi la porte des jeux mutlijoueurs. Chaque instance de *« Joueur »* doit donc préciser les types d'*« Elément »* auxquels s'adressent ses messages. Potentiellement, cela permet de créer des jeux vidéo dont les règles stipulent uniquement les « possibilités d'actions » des *« Elément »*. Libre après à chaque joueur de choisir quel type d'*« Elément »* il souhaite contrôler. Ce système nous a notamment permis de réaliser un *Pac-man* dont les joueurs peuvent choisir de contrôler indifféremment *Pac-man* ou les fantômes qui le pourchassent *(p.282)*.



99. Interaction des différentes classes mères de *Gam.B.A.S. 2.0* à travers le *modèle ISICO*

Enfin, cette modélisation de l'*« Input »* ouvre également la voie à la création d'intelligences artificielles capables de jouer à n'importe quel type de jeu. Le champ de recherche informatique du *« General Game Playing »* vise précisément cette mission. Il s'appuie pour cela sur un langage de modélisation d'un jeu : le *Game Description Language* (Love, Hinrichs, Genesereth, & Skufza, 2008). Ce langage permet de modéliser des jeux sous forme d'une collection « d'états », associés à un ensemble « règles » permettant de passer d'un « état » à l'autre. Les chercheurs de ce champ créent des programmes d'intelligence artificielle capables de jouer à des jeux formalisés dans ce langage. Ces programmes sont a priori capables de jouer à n'importe quel jeu, car ils sont conçus pour « apprendre » à y jouer à partir de l'analyse des règles de jeu rédigées avec ce formalisme. Malheureusement, le *Game Description Language* est limité à la formalisation de jeux relativement simples, en général des jeux de plateau tels que les échecs ou les dames. Dans l'espoir de créer des connexions potentielles vers le champ du *« General Game Playing »*, nous avons conçu l'architecture de cette version de *Gam.B.A.S.* de manière à ce qu'un joueur piloté par une intelligence artificielle puisse « jouer » selon des modalités identiques à celles d'un joueur humain. Pour envoyer des « messages » d'action, les instances de la classe *« JoueurIntelligenceArtificielle »* ne s'appuient pas sur les instances des classes héritant de *« Périphérique »*. Elles peuvent générer leurs « messages » en s'appuyant sur divers algorithmes d'intelligence artificielle, sachant que tous les *« Joueur »* reçoivent en paramètre

l'intégralité des *« Règle »* du jeu au démarrage de la partie. Dans la version actuelle de notre outil, la classe *« JoueurIntelligenceArtificielle »* génère tout simplement des « messages » de manière aléatoire. Mais il est facilement envisageable d'y implémenter un système de classeur et autres algorithmes génétiques. Pour faciliter le travail de la fonction de « fitness » sur laquelle repose de tels algorithmes, nous proposons une classe d'action *« EnvoiFeedbackJoueur »* qui permet d'envoyer directement une valeur numérique aux *« Joueur »* en fonction des *« Eléments »* qu'ils sont paramétrés pour contrôler. Un concepteur de jeu peut donc explicitement formuler un retour qualitatif au joueur sur ses actions, à l'image des systèmes de « score » pour les joueurs humains. Par exemple, dans notre exemple de *Pac-man* *(p.281)*, à chaque fois que le glouton jaune entre en collision avec une pastille, un feedback de valeur « +1 » est envoyé au joueur contrôle cette classe d'*« Elément »*. A l'inverse, si *Pac-man* entre en collision avec un fantôme, un feedback de valeur « -100 » est envoyé au joueur qui contrôle cette classe d'*« Elément »*. Si c'est un *« JoueurHumain »* qui contrôle *Pac-man*, nous affichons tout simplement le total des points gagnés ou perdus dans un compteur de score. S'il s'agit d'un *« JoueurIntelligenceArtificielle »*, un compteur de « score » similaire est utilisé. Mais au lieu d'être affichée, sa valeur sert de critère d'évaluation, par exemple pour la fonction de « fitness » d'un système de classeur. L'avantage de l'approche de modélisation d'un jeu vidéo proposée par l'architecture de *Gam.B.A.S.* est qu'elle permet de représenter tout type de jeu, et non pas seulement des jeux de plateau comme le *Game Description Language*. Nous espérons donc pouvoir, dans les évolutions futures du projet, l'utiliser à des fins de recherche en *« General Game Playing »*.

2.1.3.5 Gestion de la boucle principale

Dernier aspect de l'architecture générale de ce logiciel, la « boucle principale » du programme. Dans les premiers prototypes, la gestion de la boucle principale était assurée directement dans le script principal du programme. Pour cette nouvelle version nous l'avons externalisée dans une classe *« Gambas »* dédiée. A chaque cycle de calcul, cette boucle principale effectue des opérations de base, comme lire les « messages » de l'interface entrante *(p.277)* ou émettre des évènements. Ces évènements permettent par exemple à la condition *« Toujours »* d'être validée à chaque cycle de calcul. Ils déclenchent aussi l'actualisation du rendu de tous les *« Afficheur »* du jeu, ou encore permettent d'effectuer le calcul exhaustif des collisions entre objets en appelant la méthode idoine de l'*« Afficheur »* principal. En effet, un jeu peut dorénavant posséder plusieurs *« Afficheur »* ainsi que plusieurs *« Joueur »*. Cela permet par exemple de créer des jeux multijoueurs avec plusieurs écrans différents pour chaque joueur. Les listes des *« Afficheur »* et des *« Joueur »* sont stockées directement dans des propriétés de l'instance de *« Gambas »* représentant le jeu. Cette instance enregistre également la liste des *« Règle »* du jeu et la liste de tous ses *« Elément »* courants. La création et la suppression des *« Elément »* d'un jeu donné sont donc gérées directement au sein de l'instance de *« Gambas »*. Pour cela, les actions « *CréerElément »* et *« SupprimerElément »* transmettent la liste des *« Elément »* aux méthodes idoines de la classe *« Gambas »*, après avoir évaluées la liste des *« Elément »* à modifier en appelant les méthodes *« évaluer() »* de leurs paramètres encapsulés dans des instances d'*« Expression »* *(p.275)*.

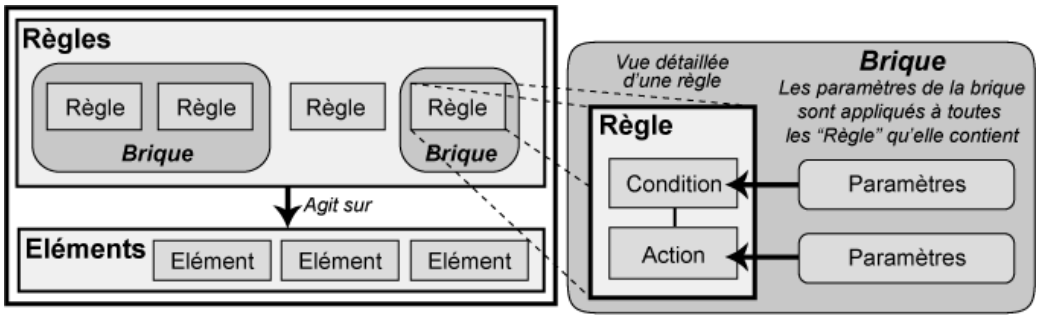
De même, l'évaluation des expressions liées aux instances de *« CiblageElement »* sont effectuées par des méthodes de *« Gambas »*. Chaque expression *« CiblageElement »* évalue tout d'abord son paramètre pour déterminer si elle doit renvoyer tous les *« Elément »* contenus dans un groupe, tous les *« Elément »* d'un type précis, ou un *« Elément »* unique. Les *« Elément »* d'un même type sont associés à la même représentation graphique *(par exemple « balle bleue »)*, alors que les éléments d'un même groupe peuvent être de types différents *(par exemple le groupe « balles » qui rassemble « balle bleue » et « balle rouge »)*. Selon le type d'éléments recherché, l'instance de *« CiblageElément »* appelle une méthode différente de *« Gambas »* pour d'obtenir la liste des *« Elément »* correspondants aux critères de ciblages précisés par le concepteur du jeu. Le fait de dissocier ainsi les types de requêtes de

ciblage sur les *« Elément »* permet d'optimiser les opérations spécifiques à chaque type de requête, notamment par un système de « mise en cache » des résultats.

Enfin, le fait d'avoir ainsi créé une classe *« Gambas »* centralisant toute la gestion d'un jeu vidéo permet de faciliter la création de plusieurs jeux vidéo différents tournants en parallèle. Il suffit simplement de créer deux instances de *« Gambas »* et de leur attribuer des *« Règle »* différentes.

2.1.3.6 Implémentation des « briques de GamePlay »

D'une manière analogue aux premiers prototypes de *Gam.B.A.S.* *(p.270)*, les *« briques de GamePlay »* sont tout simplement modélisées à travers une classe mère *« Brique »*, de laquelle héritent les différentes classes telles que *« BriqueEviter »* ou *« BriqueDéplacer »*. Le constructeur de chaque classe héritant de *« Brique »* permet de générer une liste de *« Règle »* permettant d'implémenter la brique. Ce constructeur accepte toujours des paramètres, comme les *« Elément »* ciblés par la brique, qui sont ensuite passés automatiquement à la liste des *« Règle »* créées. Cependant, l'ajout d'une approche de ciblage des *« Elément »* bien plus puissante *(p.275)* permet enfin de créer une brique telle que *« BriqueDéplacer (balle bleue) »*, qui permettra de donner des capacités de déplacement contrôlées par le joueur à toutes les instances d'*« Elément »* de type « balle bleue » *(p.272)*. Autre nouveauté découlant de la nouvelle gestion des éléments *(p.273)*, si une brique génère des *« Règle »* qui impliquent que les *« Elément »* ciblés possèdent certaines propriétés *(par exemple position X et position Y pour « BriqueDéplacer »)*, elle pourra dorénavant les ajouter dynamiquement auxdits *« Elément »*, de manière transparente pour le concepteur.



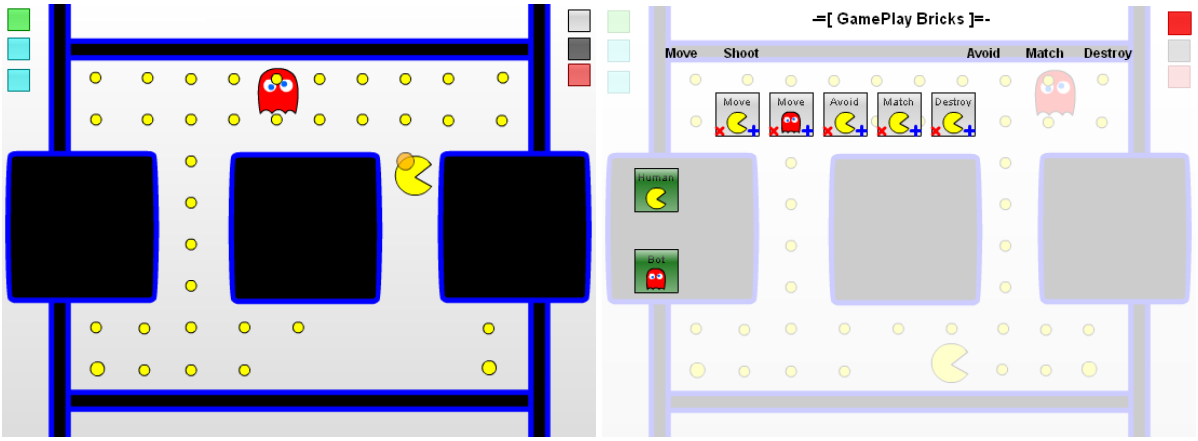
100. Modélisation des « briques » dans le prototype #3 de *Gam.B.A.S.*

Afin de pouvoir spécifier facilement les briques *(p.271)*, nous avons cette fois-ci choisi de recréer le jeu *Pac-man* avec ce nouveau prototype de notre outil. Nous avons donc créé des classes permettant d'implémenter les briques *« Déplacer »*, *« Eviter »*, *« Atteindre »*, *« Détruire »* ainsi que *« Tirer »*. En utilisant les quatre première briques, nous avons pu recréer les bases de ce grand classique du jeu vidéo : un « élément » *Pac-man* contrôlé par le joueur se déplaçant dans quatre directions tout en étant bloqué par des « éléments » murs, la destruction « d'éléments » pastilles lors de la collision entre *Pac-man* et ces dernières, et la destruction du *Pac-man* au contact avec les « éléments » fantômes. Nous avons également utilisé ces briques pour modéliser des objectifs de jeu pour un autre point de vue que celui du joueur : les « éléments » fantômes doivent essayer de détruire « l'élément » *Pac-man* tout en évitant de toucher les « éléments » murs qui les bloquent. Profitant de la notion de feedback numérique implémentée à travers les classes héritant de *« Joueur »* *(p.277)*, nous avons attribué à chaque brique la possibilité de générer un feedback. Nous avons alors créé une instance de *« JoueurHumain »*, attribuée à « l'élément » *Pac-man*, et une instance de *« JoueurIntelligenceArtificielle »*, dédiée à « l'élément » fantôme. Avec seulement cinq *« Brique »* correctement paramétrées et ces deux *« Joueur »*, nous obtenons un jeu vidéo reprenant le principe général de *Pac-man*. Le joueur déplace le glouton dans quatre directions tout en étant bloqué par les murs du labyrinthe. Il gagne des points pour chaque pastille qu'il détruit en la touchant, et meurt s'il touche un fantôme. Ce dernier est piloté par l'intelligence

artificielle, qui, à partir de l'analyse des règles de jeu, déduit que le fait d'entrer en collision avec « l'élément » *Pac-man* maximiserait son « score », alors que chaque collision avec un mur le ferait diminuer. L'intelligence artificielle basique que nous avons programmée envoie alors de manière aléatoire des « messages » d'interface entrante permettant de déplacer le fantôme *(p.277)*, tout en essayant de se souvenir quels « messages » permettent d'augmenter le plus son score total de feedback. S'il ne s'agit pas à proprement parler d'un algorithme génétique, cela reprend grossièrement l'idée générale d'un système de classeur. Il semblerait donc envisageable d'utiliser *Gam.B.A.S.* comme un terrain d'expérimentation pour créer une intelligence artificielle pouvant « apprendre » à jouer à n'importe quel jeu à partir d'un algorithme génétique se basant sur le « feedback » pour évaluer la pertinence des différentes séquences de « message d'action » qu'il génère. Mais là n'est pas l'objectif de cette expérimentation logicielle. Pour l'instant, l'intelligence artificielle basique que nous avons implémentée déplace le fantôme dans le labyrinthe pour donner plus de difficulté au joueur contrôlant le *Pac-man*. Mais rien n'empêche d'inverser les rôles, et d'attribuer *Pac-man* au *« JoueurIntelligenceArtificielle »* tandis que *« JoueurHumain »* contrôle le fantôme. Il suffit pour cela de modifier la « cible » de chaque *« Joueur »*. Et nous avons justement développé un petit *« éditeur visuel »* permettant de le faire.

2.1.3.7 Réalisation d'un « éditeur visuel »

En plus de tous ces changements d'architecture, nous avons créé un petit *« éditeur visuel »* associé au logiciel. La création des *« Règle »* par l'association d'instances de *« Condition »* et *« Action »* restent cependant basées sur l'utilisation du langage de programmation de *Flash*. Il en va de même pour la création de *« Brique »* et la spécification des *« Règle »* qu'elles génèrent. Cet *« éditeur visuel »* permet uniquement d'ajouter, modifier ou supprimer des instances de *« Brique »* à un jeu qui été créé avec *Gam.B.A.S.* en utilisant le langage de programmation. Cet éditeur permet de modifier le jeu « à la volée » *(p.264)*. En cliquant sur le bouton idoine, le jeu se met en pause et les différentes *« Brique »* qu'il utilise sont affichées. Chaque *« Brique »* est représentée par son nom, assortie d'un visuel du type d'*« Elément »* ciblé par la brique. En cliquant sur ce visuel, il est possible de changer ce type pour n'importe quel autre *« Elément »* définit dans le jeu *(la définition des éléments nécessitant elle aussi le recours au langage de programmation)*. Il est ainsi facile d'attribuer la brique *« Déplacer »* aux pastilles au lieu du *Pac-man*. Mais cela n'implique pas que le joueur contrôlant le *Pac-man* contrôle maintenant les pastilles. Non, le *Pac-man* est toujours contrôlé par le joueur. Mais comme ce type d'*« Elément »* n'a plus la capacité de se déplacer d'après les « règles » du jeu, le joueur n'a plus de possibilité d'action. *« L'éditeur visuel »* propose également de modifier, selon le même principe, les types d'*« Elément »* attribués aux deux joueurs en présence : un *« JoueurHumain »* et un *« JoueurIntelligenceArtificielle »*. Au prix d'un clic supplémentaire, il est donc possible que le *« JoueurHumain »* contrôle dorénavant les pastilles. Mais en soit, le jeu n'a plus trop d'intérêt puisque aucune *« brique de GamePlay »* n'a été utilisée pour définir un objectif à ces pastilles. Un clic supplémentaire permet alors d'ajouter, par exemple, une brique *« Tirer »* aux pastilles. *« L'éditeur visuel »* permet de configurer les paramètres supplémentaires de la brique, et donc de définir que les pastilles tirent dorénavant des *« Elément »* de type *Pac-man*. Ajoutons également une brique *« Atteindre »* et une brique *« Détruire »* attribuées au *Pac-man*, avec comme paramètres supplémentaires le fantôme comme cible à détruire pour recevoir un feedback positif. Notre jeu de *Pac-man* se trouve alors complètement transformé : le joueur humain contrôle dorénavant les pastilles, qui à chaque clic de la souris tirent un *Pac-man* dans l'espoir de toucher le fantôme pour le détruire.



101. *Gam.B.A.S.* : prototype #3 : écran de jeu et « éditeur visuel »

2.1.3.8 Bilan de cette expérimentation

Si elle n'offre toujours pas *« d'éditeur visuel »* permettant la création de nouveaux jeux vidéo autonomes, cette nouvelle version de *Gam.B.A.S.* propose un outil permettant de modifier, « à la volée » *(p.264)*, les « règles du jeu » à travers la manipulation de *« briques de GamePlay »*. La création de tous les autres aspects du jeu s'appuie sur l'utilisation d'un langage de script propriétaire, celui de *Flash*. En l'état, *Gam.B.A.S.* s'apparente donc plus à un « framework » qu'à un véritable logiciel auteur. Pourtant, son architecture, qui modélise la structure d'un jeu d'une manière proche de la pensée d'un Game Designer *(p.258)*, laisse entrevoir la possibilité de continuer le développement de son *« éditeur visuel »* pour permettre la création de toutes les composantes d'un jeu vidéo. Cette architecture propose des classes d'objets pour les *« Elément »*, les *« Règles »* composées de *« Condition »* et *« Action »* avec des paramètres s'appuyant sur des *« Expression »*, les *« Joueur »* et leur *« Périphérique »* d'interface entrante, les *« Afficheur »* gérant l'interface sortante sans oublier les diverses *« Brique »*. Les différentes composantes d'un jeu vidéo étant formalisées au sein de classes d'objets distinctes, la conception d'un *« éditeur visuel »* permettant de manipuler ces différentes classes semble être la prochaine étape pour réaliser une véritable usine à jeux assortie d'un *« éditeur visuel »* puissant *(p.260)*.

Cependant, cette version de *Gam.B.A.S.* possède toujours des limites, et non des moindres. Tout d'abord, la performance technique du logiciel pour la création d'un jeu est assez faible. Un jeu construit avec de nombreuses *« Règle »* et *« Elément »* tournera trop lentement pour envisager l'utilisation de ce moteur pour la production de Serious Games comparables à ceux créés par nos étudiants *(p.243)*. Certes, il ne s'agit là que d'un prototype réalisé sans aucune volonté particulière d'optimisation. Il est donc tout à fait imaginable de retravailler le code pour le rendre plus performant. Mais nous pensons que ces chutes de performances sont principalement dues à notre choix d'architecture, et plus précisément la manière dont fonctionne le « moteur d'expression » *(p.275)* et celui des « règles » *(p.274)*. Si le choix de créer des classes séparées pour chaque condition, action et même expression est pertinent d'un point de vue théorique, dans la pratique cela entraîne la création d'un nombre conséquent « d'objets » pour la création des « règles ». A l'usage, le fait d'avoir à créer une nouvelle instance d'objet pour générer un nombre aléatoire, voire pour passer une simple chaîne de texte en paramètre, s'avère plutôt besogneux. Il nous semblerait plus pertinent, autant pour des raisons de performance que de simplicité d'utilisation, de ne pas avoir recours à la création de classes spécifiques à chaque condition, action ou expression. La création d'une seule classe pour chaque type d'élément, qui pourrait ensuite recevoir en paramètre le nom de la condition, action ou expression correspondante, permettrait par exemple d'alléger considérablement cette architecture.

Enfin, cette nouvelle version de *Gam.B.A.S.* ne résout toujours pas la problématique de la « spécification » des briques. Puisqu'il faut créer des briques en pensant à un nombre restreint de genres vidéoludique, il faudrait également proposer un *« éditeur visuel »* permettant au concepteur de « créer ses propres briques », au lieu de le limiter à celles qui ont été programmées au sein de l'outil. Le fait de permettre à un utilisateur de créer lui-même ses propres « briques » est la fonctionnalité sur laquelle repose la possibilité de réaliser un outil alliant la puissance des usines à jeux « généralistes » avec la simplicité d'utilisation de celles qui sont « spécialisées » *(p.261)*. Loin d'être anecdotique, un tel changement d'approche implique de repenser une bonne partie de l'architecture de notre outil.

2.1.4 CONCLUSION DU PROJET GAM.B.A.S.

Les trois prototypes du projet *Gam.B.A.S.* représentent notre première tentative de réalisation d'un outil technique. S'il elle n'a pas permis d'aboutir à un outil complet et fonctionnel, cette expérimentation logicielle aura été riche d'enseignements. En recoupant ces enseignements avec les autres travaux et outils étudiés dans cette thèse, nous avons pu définir le « cahier des charges » d'un outil idéal pour le contexte de nos cours *(p.257)*. Pour autant, *Gam.B.A.S.* répond déjà à certains points de ce « cahier des charges ». Nous pourrions alors imaginer continuer le développement de son dernier prototype pour essayer d'implémenter les points manquants. Malheureusement, comme nous l'avons déjà évoqué *(p.283)*, l'architecture de *Gam.B.A.S.*, si elle implémente bien un outil théorique en la forme des *« briques de GamePlay »* *(p.263)*, n'est pas du tout adaptée pour permettre à l'utilisateur de créer lui-même ses « briques » par un *« éditeur visuel »* *(p.261)*. Si l'on ajoute la nécessité d'optimiser cette architecture pour obtenir de meilleures performances techniques, il semble plus pertinent de repartir sur une nouvelle base, plutôt que de tenter de corriger le dernier prototype réalisé.

Nous ne poursuivrons donc pas le projet *Gam.B.A.S.*, mais utiliserons ses nombreux enseignements et ses meilleures idées pour tenter d'aboutir à la création d'un outil technique véritablement utilisable « sur le terrain » pour créer des Serious Games.

2.2 LE PROJET « LUDOFORGE »

Commencé en 2009, le projet *LudoForge* vise à créer un logiciel répondant à tous les points formalisés dans le cahier des charges *(p.257)*. S'il reprend certaines des idées du projet *Gam.B.A.S.*, *LudoForge* reste un projet partant d'une base vierge. Au-delà d'une nouvelle architecture, nous avons également changé le langage de programmation utilisé pour le réaliser, dans l'espoir de pallier les faibles performances techniques constatées sur *Gam.B.A.S.* *(p.283)*. Si nous restons sur la technologie *Flash* pour répondre à la dimension « Jeu 2.0 » du projet *(p.264)*, nous abandonnons le langage *Actionscript 2* pour passer à l'*Actionscript 3*. Plus qu'une simple évolution, ces deux langages de programmation, incompatibles entre eux, sont basés sur une architecture différente. Afin d'offrir de meilleures performances, *Actionscript 3* s'inspire largement l'architecture de *Java*, s'appuie sur des routines d'affichages graphiques réécrites, et permet entre autres de typer de manière stricte les variables pour gagner en performance. Comme nous l'avons déjà mentionné *(p.266)*, la réalisation d'un nouvel outil technique de création vidéoludique est une tâche de longue haleine. En parallèle aux nombreux autres projets liés au *CIFRE* ainsi qu'au travail de recherche théorique, cette expérimentation logicielle ne pouvait vraisemblablement pas aboutir à un outil complet répondant à l'ensemble du cahier des charges durant cette thèse. Nous nous sommes alors fixé un objectif intermédiaire à atteindre à la fin de la thèse : la réalisation d'un « moteur d'interprétation » suffisamment complet et fonctionnel pour permettre la création de Serious Games *(p.257)*. Une fois cette première étape franchie, il sera possible d'envisager la réalisation d'une « interface auteur » *(p.293)* dans les éventuelles suites du projet après la thèse. Nous détaillons ici l'architecture de ce « moteur d'interprétation », ainsi que son utilisation pour la création de jeux vidéo et Serious Games.

2.2.1 ARCHITECTURE GENERALE

Cette section détaille brièvement les principales caractéristiques de l'architecture de *LudoForge*, qui reprend de nombreux concepts de *Gam.B.A.S. 2.0*, bien que leurs codes sources n'aient rien en commun. De plus, comme l'illustrent les diagrammes de classes respectifs de ces deux projets *(p.318 et p.315)*, l'architecture de *LudoForge* a volontairement été conçue pour être plus légère que celle de *Gam.B.A.S. 2.0*.

2.2.1.1 Héritages du projet « Gam.B.A.S. »

De nombreuses fonctionnalités de *LudoForge* sont héritées du dernier prototype de *Gam.B.A.S. (p.273)*. Par exemple, l'architecture générale du logiciel reste pensée pour permettre au concepteur de modifier les règles du jeu « à la volée », afin de faciliter le travail de prototypage *(p.264)*. Mais surtout, cette architecture conserve une modélisation séparée des composantes *« Input »*, *« Compute »* et *« Output »* du *modèle ISICO (p.277)*. Du côté de l'interface sortante, l'idée d'utiliser une classe mère *« Afficheur »* de laquelle héritent plusieurs moteurs de rendus a été conservée. La classe *« Rendu2D »* a par contre été réécrite à des fins d'optimisation. De nouvelles fonctionnalités lui ont également été rajoutées, telle que la possibilité de gérer des « calques » de *« Sprite »* de manière dynamique, la gestion du scrolling ou encore des modes de détection de collision par « boites » ainsi que « au pixel près ». Une classe *« RenduSonore »* a été introduite pour permettre à l'interface sortante de *LudoForge* de gérer du son en plus des graphismes.

En ce qui concerne l'interface entrante, nous avons là aussi conservé l'idée d'utiliser une classe mère *« Joueur »* de laquelle héritent *« JoueurHumain »* et *« JoueurIntelligenceArtificielle »*. Cependant, afin de réduire le nombre d'instances utilisées pour modéliser un jeu *(p.283)*, nous avons supprimé les classes *« Périphérique »*. Seul le *« JoueurHumain »* ayant besoin de pouvoir lire l'état des périphériques d'entrée standard *(clavier et souris)*, nous avons intégré les méthodes dédiées à ces tâches directement à l'intérieur de cette classe. Autre nouveauté, les *« Joueur »* peuvent maintenant posséder des propriétés pour enregistrer leur « état courant », d'une manière analogue à la façon dont les *« Elément »* étaient gérés pour *Gam.B.A.S. (p.273)*. Cela implique la disparition de la notion de « feedback » comme seule variable affectée au joueur. De même, la notion de type d'*« Elément »* ciblé par un joueur a été supprimée. En effet, le concepteur du jeu est dorénavant libre de créer les « propriétés » qu'il souhaite au sein du *« Joueur »*, que ce soit des variables numériques, textuelles, scalaires ou des références à des *« Elément »*.

Cet ajout de propriétés dynamiques à la classe *« Joueur »* est également lié à la gestion de la boucle principale. Nous avons conservé le principe d'avoir une classe qui « contienne » la gestion de cette boucle *(p.280)*, mais nous l'avons cette fois appelée *« Gameplay »*. En effet, de nombreux jeux vidéo sont composés de plusieurs « gameplay » différent. Par exemple, un même jeu pourra proposer des passages de type « jeu de plateforme » entrecoupés de phases dans lesquelles le joueur déplace un avatar sur une carte pour sélectionner un niveau *(cf. par exemple New Super Mario Bros. Wii)*. S'il s'agit là d'un même jeu vidéo, ces deux parties semblent pourtant reposer sur des « règles de jeu » suffisamment différentes pour qu'il soit plus simple de les modéliser séparément. Ainsi, un « jeu » créé avec *LudoForge* pourra être composé de plusieurs *« Gameplay »* qui obéissent à des règles différentes. Concrètement, chaque *« Gameplay »* contient une collection d'instances de *« Règle »*, *« Elément »*, *« Joueur »* et *« Afficheur »* pour permettre la modélisation d'un principe de jeu donné. Rien n'empêche alors le concepteur du jeu de n'utiliser qu'un seul *« Gameplay »* pour proposer un jeu simple basé sur un seul « principe de jeu ». Mais il peut aussi bien utiliser plusieurs *« Gameplay »* pour proposer des phases de jeu différentes. Si besoin, il est même envisageable de créer plusieurs instances de *« Gameplay »* tournant en parallèle. Pour ce premier prototype, nous utilisons les fonctions de scénario interne à *Flash* pour naviguer entre les différents *« Gameplay »*. Mais nous envisageons la création d'une classe *« Jeu »* destinée

à gérer les instances de *« Gameplay »* pour les prototypes suivants. Si les différentes instances de *« Gameplay »* possèdent des instances de *« Règle »*, *« Elément »* et *« Afficheur »* qui leurs sont propres, une même instance de *« Joueur »* peut par contre être assignée à plusieurs *« Gameplay »*. C'est même le seul moyen de passer des variables entres les différents *« Gameplay »* : en utilisant les propriétés dynamique de *« Joueur »*. Là encore, cette modélisation reflète le mode de pensée d'un Game Designer *(p.258)*, pour lequel un même « Joueur » sera amené à tester plusieurs phases de jeu, qui correspondent à des *« Gameplay »* différent durant une partie.

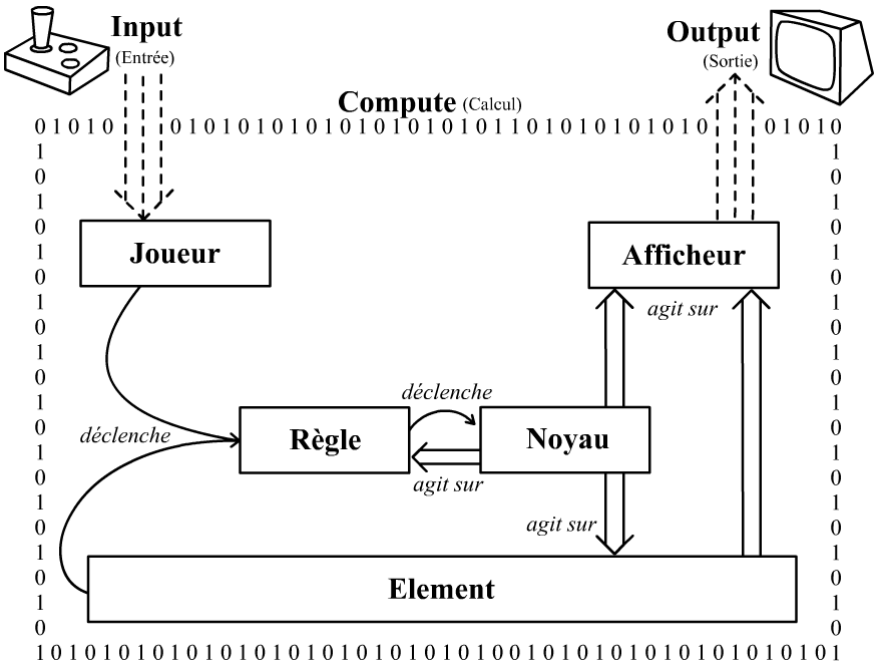
Enfin, dernier héritage de *Gam.B.A.S.*, la gestion des « éléments » reposant sur une unique classe *« Elément »* pouvant représenter différents « types d'éléments » à travers des propriétés dynamiques. Ces dernières peuvent toujours être soit un nombre *(float)*, soit un texte *(string)*, soit une liste *(array)* ou bien une référence à un *« Elément » (p.273)*. Cet aspect est resté sensiblement le même, y compris pour la manière dont les *« Règle »* délèguent le ciblage des éléments à la boucle principale. Au-delà de cet aspect, la gestion des « règles » a été complètement repensée, comme expliqué ci-après.

2.2.1.2 Gestion des « règles »

Dorénavant, les règles sont uniquement modélisées à partir d'une classe *« Règle »*. Afin d'éviter le trop grand nombre d'instances générées par l'architecture des nombreuses classes héritant de *« Condition »* et *« Action » (p.274)*, nous sommes partis dans une autre direction. La classe *« Règle »* possède trois listes comme propriétés : *« conditions »*, *« actions »*, *« actionsElse »*. Ces listes sont destinées à accueillir des tableaux associatifs contenant des paramètres permettant de préciser quelles sont les « conditions » et « actions » composant la « règle ». Le code des méthodes liées aux diverses « conditions » et « actions » utiles à la création d'un jeu vidéo *(détection des messages de l'interface entrante, comparaison et modifications des propriétés des « Elément », création et destruction des « Elément », envoi de messages à l'interface sortante…)* est désormais intégré à une classe *« Noyau »*. Cette classe *« Noyau »* ne possède que des méthodes statiques. Elles sont divisées en deux catégories selon leurs paramètres d'entrée et le type de variable qu'elles retournent : les méthodes « condition » et les méthodes « action ». En plus de ses paramètres spécifiques, une méthode « condition » attend une liste d'*« Elément »* comme paramètre d'entrée, et renvoie une liste d'*« Elément »* en retour. Ainsi, lorsqu'une *« Règle »* souhaite vérifier une de ses conditions, elle va tout simplement appeler la méthode statique correspondante de la classe *« Noyau »*, en lui passant éventuellement une liste d'*« Eléments »* comme paramètre. Dans tous les cas, cette méthode renverra la liste des *« Eléments »* vérifiant la condition. Ainsi, s'il y a plusieurs conditions dans la *« Règle »*, cette dernière passera la liste renvoyée par sa première condition à sa deuxième condition, puis celle renvoyée par sa deuxième condition à sa troisième, et ainsi de suite. Une fois toutes les conditions vérifiées, la liste d'*« Elément »* renvoyée par la dernière condition sera successivement passée à toutes les méthodes statiques de la classe *« Noyau »* référencée dans la liste *« actions »* de la *« Règle »*.

La notion de condition « déclencheur » a donc été complètement abandonnée, de même que le recours à la programmation évènementielle pour la gestion des *« Règle » (p.274)*. Désormais, la boucle principale appelle successivement la méthode *« appliquer() »* de chaque *« Règle »*. Cette méthode va alors appeler par référence les méthodes condition du *« Noyau »* qui sont référencées dans la propriété *« conditions »* de la *« Règle »*, accompagnée par les paramètres spécifiques à chaque méthode *(nom des propriétés à comparer, types d'« Elément » dont les collisions doivent être vérifiées…)*. Si toutes les conditions sont validées, la *« Règle »* appellera par référence les méthodes action du *« Noyau »* référencées dans sa propriété *« actions »*. Si les conditions ne sont pas validées, elle appellera les éventuelles méthodes action du *« Noyau »* référencées dans la propriété *« actionsElse »*. Il n'est donc plus nécessaire de créer des instances d'objets pour représenter les conditions et actions d'une

« Règle » et leurs paramètres. Toutes ces informations sont stockées dans des listes de tableaux associatifs, ce qui allège grandement l'architecture du logiciel. Cela permet d'une part de gagner en performance, mais simplifie également de manière considérable la tâche d'écriture de règles. Un concepteur utilisant *LudoForge* doit écrire les *« Règle »* de son jeu à partir du langage de programmation de *Flash*, puisque nous ne réalisons pour l'instant qu'un « moteur d'interprétation » dépourvu d'« interface auteur ». Le fait de ne plus avoir à créer de nombreuses instances pour la rédaction des *« Règle »* lui permet donc de gagner du temps. Cette nouvelle architecture de gestion des *« Règle »* présente un autre avantage : en plus des tableaux associatifs, les propriétés *« actions »* et *« actionsElse »* peuvent également contenir des *« Règle »*. Il est ainsi possible de créer sans limite des *« Règle »* encapsulées, ouvrant de fait la voie à la création de jeux vidéo aux systèmes de « règles » riches et complexes.



102. Interaction des différentes classes mères de *LudoForge* à travers le *modèle ISICO*

Selon cette nouvelle architecture, pour créer de nouvelles conditions ou actions il suffit de rajouter une méthode statique à la classe *« Noyau »* tout en respectant le formalisme établi. Afin de faciliter les évolutions de cette véritable « bibliothèque centralisée » d'actions et conditions, nous avons créé d'autres classes analogues à *« Noyau »* qui contiennent des actions et conditions plus spécifiques. Par exemple, la classe *« Mouvement2D »* contient toutes les conditions et actions nécessaires à la création de « règles de jeu » liées au déplacement d'*« Elément »* dans un espace en deux dimensions *(détection des collisions…)*. D'autres classes ont permis l'intégration des *API* de *Google Analytics* et *Playtomic* *(p.134)*, indispensables au suivi des joueurs *(p.264)*. Mais les différentes méthodes statiques de ces classes « bibliothèque » ne contiennent pas que des conditions et des actions, elles rassemblent également des méthodes associées au nouveau « moteur d'expression ».

2.2.1.3 Un « moteur d'évaluation d'expression » plus puissant

Dans *Gam.B.A.S.*, la création « d'expressions » permettant aux paramètres des *« Règles »* d'être évalués à chaque cycle de calcul était assurée par une collection de classes héritant d'*« Expression »* *(p.275)*. Chacune de ces classes contenait donc une partie du code permettant au logiciel d'interpréter les expressions imaginées par le concepteur du jeu. En plus d'alourdir les performances globales, ce mode de fonctionnement était relativement complexe à manipuler. Selon le type de paramètres souhaités, il fallait utiliser une classe différente *(« Nombre », « Nombre Aléatoire »…)*. Mais surtout, il fallait créer une nouvelle instance pour chaque paramètre, y compris pour une simple chaîne de texte *(p.283)*.

LudoForge est donc basé sur une approche différente, qui simplifie et optimise considérablement cet aspect. Concrètement, les *« Règles »*, et toutes les méthodes conditions et actions des « bibliothèques » auxquelles elles font référence, appellent une méthode *« évaluer() »* avant d'utiliser leurs paramètres. Située dans la classe *« Noyau »*, cette méthode statique centralise le code dédié à l'interprétation des « expressions » rédigées par le concepteur dans les paramètres des *« Règle »*. Cette méthode attend comme paramètre d'entrée des « données » à traiter, ainsi que le « type » des données qui lui ont été transmises *(nombre, texte, liste ou référence à un « Elément »)*. Elle renvoie en retour une liste contenant les données résultant de son évaluation, ainsi qu'une éventuelle liste d'*« Elément »* à laquelle correspondent les données. En effet, il peut dorénavant arriver qu'une expression renvoie plusieurs paires « données – liste d'éléments ». Par exemple, si l'expression rédigée par le concepteur du jeu demande la distance entre les éléments « balle bleue » et « balle rouge », cette méthode renverra de manière exhaustive la distance entre chaque paire de « balle bleue » et « balle rouge » existant dans le jeu. Le fait d'associer chaque distance aux *« Elément »* permet, par exemple, de retrouver uniquement la distance entre la première « balle rouge » créée et la « balle bleue » sur laquelle a cliqué le joueur.

A cette fin, ce nouveau « moteur d'expression » permet au concepteur d'effectuer des calculs complexes à partir de valeurs littérales ou de propriétés provenant d'éléments résultant d'une « expression de ciblage ». Mais ce moteur d'expression peut également, dans ses calculs, utiliser des données provenant d'autres méthodes de calculs d'expression. En effet, nous avons défini dans *« Noyau »*, en plus du formalisme caractérisant les conditions et les actions, un formalisme définissant les paramètres d'entrée et valeurs de retours d'une méthode statique utilisable par le moteur d'expression. La méthode *« évaluer() »* répondant à ce formalisme, elle peut s'appeler elle-même de manière récurrente au besoin. Mais un concepteur utilisera plutôt ce système pour appeler d'autres méthodes statiques permettant d'effecteur des calculs spécifiques, comme les calculs de distances entre *« Elément »*, les calculs de sinus, cosinus ou tangente sur des valeurs, la génération de valeurs aléatoires… La méthode *« évaluer() »* analyse l'expression textuelle rédigée par le concepteur du jeu, et décide le cas échéant d'appeler une autre méthode de calcul d'expression. Elle traitera alors le résultat de cette dernière pour le renvoyer dans un format qui sera attendu par les méthodes statiques de type « conditions » et « actions » référencées dans les *« Règle »*.

Pour permettre à *« évaluer() »* de déterminer quand récupérer les propriétés contenues dans des *« Elément »*, quand appeler une autre méthode de calcul d'expression, ou encore quand l'expression rédigée est une valeur littérale, nous avons mis en place ce que l'on pourrait considérer comme un *« langage de script propriétaire »* basique. Par exemple, une expression de la forme *« {cible.propriété} »* correspond à la récupération de propriétés sur des *« Elément »*, alors que *« [bibliothèque.méthode(nomParamètre:valeurParamètre)] »* permet de faire appel à n'importe quelle autre méthode statique de calcul d'expression contenue dans une « bibliothèque ». Pour distinguer les différentes membres d'une expression nécessitant des calculs *(par exemple pour multiplier par 2 la valeur d'une propriété d'un « Elément »)*, le concepteur du jeu doit utiliser le symbole « | » comme séparateur. La contrepartie de la puissance de ce nouveau « moteur d'expression » est que le concepteur doit pour l'instant apprendre et respecter ce formalisme basique pour la rédaction de ses expressions. Mais cet aspect pourra bien évidement être « masqué » de manière très simple par la création future d'une « interface auteur » *(p.293)*. Et quoi qu'il en soit, ce système reste bien plus simple à utiliser que la création de la grande quantité d'instances requise par *Gam.B.A.S. 2.0*, d'autant plus que le moteur d'expression de *LudoForge* est bien plus puissant.

Enfin, la gestion des comparaisons et modifications entre les différents types de variable n'est plus associée à une classe d'objet de type « interface à implémenter » *(p.275)*, mais tout simplement intégrée à deux méthodes statiques de *« Noyau »*. La méthode condition

« comparer() » permet d'effectuer des comparaisons entre les quatre types de variables reconnues par *LudoForge* : *nombre, texte, liste et référence à un « Elément »*. De son côté, la méthode action *« modifier() »* permet d'effectuer diverses opérations *(ajout, suppression, concaténation…)* sur les propriétés d'une liste d'*« Elément »*. Bien évidement, les paramètres de ces méthodes sont auparavant passés à la méthode *« évaluer() »*, ce qui permet de créer des règles de jeu aux paramètres dynamiques. Par exemple, une règle dont la « condition » compare les propriétés d'*« Elément »* à des résultats de calculs tels que la distance entre deux *« Elément »*. Ou encore une règle dont « l'action » est de modifier les propriétés d'un *« Elément »* donné suite à des opérations mathématiques effectuées sur les propriétés d'autres *« Elément »*.

En centralisant ainsi le fonctionnement du moteur d'expression de *LudoForge*, celui-ci gagne en simplicité d'utilisation tout en ouvrant les possibilités créatives du concepteur. Il est également bien plus optimisé que celui de *Gam.B.A.S.*. Mais surtout, le fait de proposer ainsi des « conditions » et « actions » de relativement « bas niveau » *(comparaison et modification de variables, création et suppression d'« Elément», communication directe avec les interfaces entrante et sortante…)* permet d'anticiper la réalisation d'une « interface auteur » permettant au concepteur d'avoir un contrôle très fin du jeu pour « créer ses propres briques » *(p.293)*.

2.2.2 UTILISATION DE LUDOFORGE POUR LA CREATION DE SERIOUS GAMES

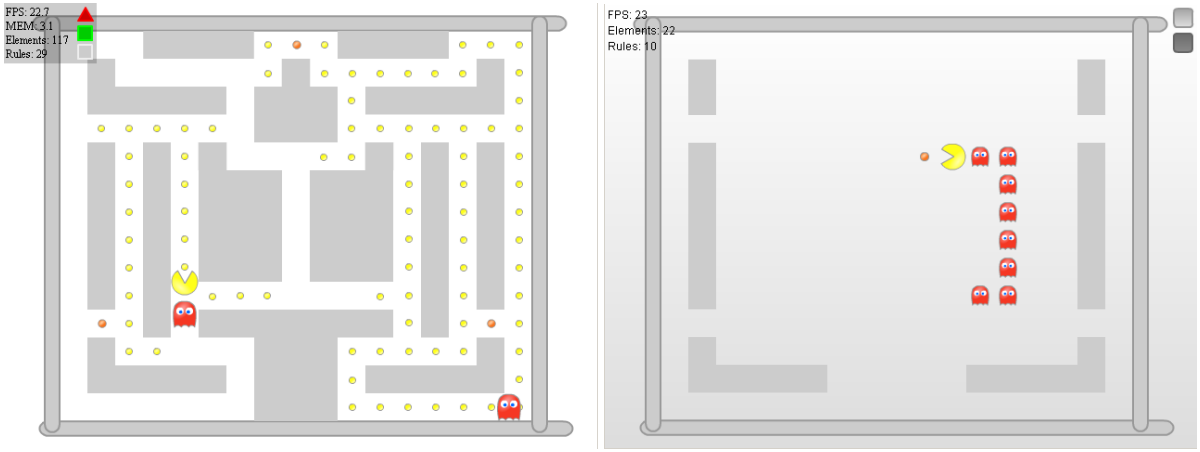
Ces différents choix d'architecture ont permis de programmer un « moteur d'interprétation » qui semble applicable à la création de Serious Games. Afin d'utiliser ce « moteur d'interprétation » pour la création vidéoludique, nous avons programmé plusieurs méthodes statiques de « conditions », « actions » et « expressions » au sein de quatre « bibliothèques ». Ces « bibliothèques » sont destinées à faciliter la création de *« Règle »* permettant la réalisation de « véritables jeux vidéo », et donc de Serious Games.

2.2.2.1 Pac-man et le jeu du serpent

Afin de tester la viabilité de cet outil en cours d'élaboration, nous l'avons utilisé pour créer deux jeux vidéo qui avaient déjà été esquissés par les différents prototypes de *Gam.B.A.S* : *Pac-man* et le *jeu du serpent*.

En ce qui concerne *Pac-man* nous avons facilement été en mesure de reproduire le jeu tel qu'il était dans le dernier prototype de *Gam.B.A.S* *(p.281)*. Mais, grâce à la gestion des « règles » et des « expressions » bien plus avancée de *LudoForge*, nous avons également été en mesure d'implémenter les nombreuses règles manquantes de la version *« Gam.B.A.S »* de *Pac-man*. Par exemple, le fait que les fantômes chassent le joueur en temps normal mais le fuient dès que ce dernier mange une « super-pastille » a été rajouté, de même que la possibilité de manger les fantômes pendant un temps limité après avoir ingéré cette même « super-pastille ». Pour cela, les fantômes ne sont plus contrôlés par une instance de *« JoueurIntelligenceArtificielle » (bien que cette option reste possible)*, mais directement par des *« Règle »* que nous avons écrites pour essayer de reproduire le comportement des fantômes du jeu originel. De plus, le fait de ne plus utiliser des « briques » imposant un mouvement générique nous a permis de recréer un mode de déplacement fidèle au jeu d'origine. Dorénavant, le joueur peut laisser une touche de direction appuyée, *Pac-man* ne tournera que lorsqu'il se trouvera à un croisement du labyrinthe. Dans la version *Gam.B.A.S.*, l'avatar du joueur tournait dès l'appui sur une touche, quitte à se bloquer contre un mur. Loin d'être anecdotique, ce genre d'ajustement très fin au sein des règles du jeu conditionne la qualité de l'expérience ludique finale du joueur *(p.149)*. Dans le cas de *Pac-man*, cela permet au joueur de « préparer » ses mouvements à l'avance, rendant le jeu bien plus agréable à manipuler.

En plus de la liberté créative accrue qu'offre *LudoForge* au concepteur, ce premier test met en lumière un autre de ses avantages par rapport à son prédécesseur : la performance. Nous avons pu sans problème réaliser un grand labyrinthe comprenant pratiquement 150 pastilles *(contre seulement 45 auparavant)* et 2 fantômes *(contre un seul auparavant)* sans que le jeu ne montre de chute de performance. Afin de pouvoir construire facilement le labyrinthe de ce remake de *Pac-man*, nous avons programmé une méthode permettant d'utiliser « l'interface auteur » de *Flash* comme *« éditeur de niveau »* pour *LudoForge*. Il suffit donc de placer les différents *« Sprite »* composant visuellement le niveau de jeu sur la scène de *Flash*. Au démarrage du logiciel, ce dernier analysera cette composition graphique pour la traduire en *« Règle »* permettant de générer un véritable niveau de jeu composé d'*« Elément »*.



103. Les jeux vidéo *Pac-man* et le *jeu du serpent* réalisés avec *LudoForge*

Après la création de ce remake de *Pac-man*, nous avons tenté de reproduire le *jeu du serpent* *(p.271)*. Là aussi, *LudoForge* s'avère bien plus pertinent que son prédécesseur pour réaliser un tel jeu vidéo. Nous avons réussi à créer des règles de mouvement mimant exactement la façon dont se déplace le serpent dans le jeu originel *(case par case)*, là où notre expérimentation précédente comportait de nombreuses approximation à ce niveau. Le jeu ainsi obtenu tourne de manière très fluide, et fut créé en seulement quelques heures. Au final, nous avons réussi à recréer *Pac-man* en utilisant seulement 29 *« Règle » (sans compter les règles imbriqués)*, et 17 pour le *jeu du serpent*. Si la création des règles nécessite toujours l'utilisation du langage de programmation de *Flash*, le fait de pouvoir créer un jeu en définissant simplement les propriétés des différents types d'*« Elément »*, puis en créant les *« Règle »* en spécifiant uniquement les noms et paramètres des « conditions » et « actions » sous forme textuelle, nous semble relativement simple d'emploi. Si pour le contexte de nos cours la présence d'un *« éditeur visuel »* reste indispensable, le « framework » que représente *LudoForge* en l'état nous semble déjà posséder un certain potentiel pour simplifier la tâche d'un Game Designer professionnel cherchant à prototyper rapidement ses idées. Nous l'avons donc utilisé en ce sens pour la création d'un Serious Game dans le cadre du *CIFRE*.

2.2.2.2 Le Jardinier Ecolo

La puissance de création que semblait offrir *LudoForge*, tout en conservant des performances techniques honorables, nous a encouragé à l'utiliser pour réaliser un « véritable » Serious Game. Nous avons employé *LudoForge* pour réaliser *Le Jardinier Ecolo*, un Serious Game dont la conception a longuement été présentée dans un précédent chapitre *(p.123)*. Cette expérience fut un véritable test « grandeur nature », que nous considérons comme réussi. En effet, *LudoForge* nous a permis de réaliser un Serious Game tournant parfaitement, qui fut d'ailleurs diffusé au grand public sur Internet. Mais au-delà du simple fait qu'il est possible de créer un « véritable jeu » avec cet outil, son intérêt vient de la manière dont il nous a aidé durant le processus de création du *Jardinier Ecolo*. Les très longues phases d'ajustement et d'affinage des différentes variables du jeu *(coût d'achat des éléments, impact des plantes sur*

le public, système de score…) ont été grandement facilitées par cet outil *(p.125)*. Le fait de visualiser l'intégralité des « règles du jeu » sur une seule page de script simplifiait grandement leur équilibrage, l'outil présentant ces « règles » du manière très proche de celle dont nous les avions imaginées « sur papier » *(p.258)*.

Afin de pouvoir évaluer l'impact de ce Serious Game « intrinsèque » en nous appuyant sur des techniques de suivi du joueur, nous avons utilisé les « bibliothèques » contenant des conditions et actions permettant de contrôler les outils génériques *Google Analytics* et *Playtomic* *(p.286)*. Il a été ainsi relativement simple d'ajouter le suivi du joueur au *Jardinier Ecolo* : il nous a suffit de rajouter des actions *« envois de données à Google Analytics et Playtomic »*, avec comme paramètre la variable du jeu qui nous intéressait *(argent du joueur, son score…)*, au sein des règles de jeu déjà existantes. Nous avons utilisé un procédé similaire pour inclure le « tutorial » au sein du jeu *(p.126)*. Nous avons tout d'abord créé un *« Elément »* dont la représentation contenait les différents « écrans » du tutorial. Puis, nous avons ajouté aux *« Règle »* existantes des conditions et actions modifiant l'apparence de cet *« Elément »* selon la « phase » du tutorial dans laquelle se trouvait le joueur. Il nous a été ainsi possible d'implémenter un premier tutorial interactif au sein même du jeu en moins de deux heures. Il fut également aisé de modifier ce tutorial suite aux retours de nos testeurs. Le gain de temps et la facilité de modification de règles du jeu ainsi modélisé dans *LudoForge* a grandement bénéficié aux nombreux cycles de prototypages et d'évaluation du jeu. *Le Jardinier Ecolo* reposant sur un principe de jeu plus complexe que celui de *Pac-man*, 53 *« Règles »* ont été nécessaires à sa création *(sans compter les nombreuses « Règles » imbriquées)*. De plus, ce Serious Game possède du son et des graphismes richement animés, contrairement à nos expériences précédentes. Et pourtant, ce jeu vidéo a fait preuve de performances techniques tout à fait acceptables lors de nos nombreux tests qualitatifs *(p.132)*. Au final, la création du *Jardinier Ecolo* nous laisse imaginer que le « moteur d'interprétation » réalisé pour *LudoForge* représente une base prometteuse pour la création d'un outil technique complet destiné à la création de Serious Games.

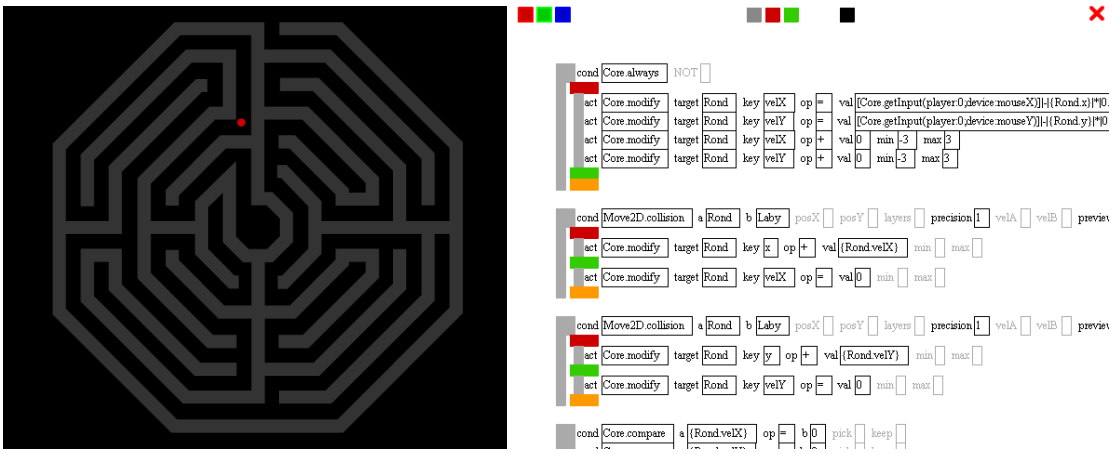
2.2.2.3 Mahjian

Nous avons également créé un quatrième jeu vidéo, certes plus modeste, en utilisant *LudoForge*. Dans le cadre du *CIFRE* au sein de la ludothèque du centre culturel ***Odyssud***, nous avons été amené à participer à la réalisation d'un jeu dit en « réalité alterné ». Généralement désigné par l'acronyme anglophone *« **A**lternate **R**eality **G**ame »*, ce type de jeu se caractérise par son côté « événementiel » et par le fait qu'il s'appuie simultanément sur plusieurs médias. Un *ARG* repose sur une fiction à laquelle participent les joueurs : *complot à déjouer, meurtre à résoudre…* Une équipe d'animateurs se mobilise pour mettre en place des vidéos, sites Internet, journaux ou même de manifestations ponctuelles visant à mêler cette fiction à la réalité. L'objectif recherché est d'immerger les joueurs dans l'histoire du jeu en utilisant plusieurs médias faisant partie de leur quotidien. Par exemple, certains personnages du jeu pourront être incarnés par de véritables acteurs que les joueurs pourront appeler ou rencontrer lors d'évènements. Le jeu est ensuite rythmé par l'apparition de nouveaux éléments de l'histoire racontée par les animateurs, jusqu'au dénouement de la partie. Un *ARG* se déroule donc sur un laps de temps clairement défini. Par exemple, un *ARG* se déroulant sur huit semaines proposera aux joueurs une nouvelle énigme à résoudre par semaine. Les joueurs doivent alors utiliser les nombreux indices disséminés dans la réalité par les animateurs du jeu pour résoudre cette énigme de manière collective. Une fois l'énigme résolue, un nouveau chapitre de l'histoire est raconté aux joueurs, puis une nouvelle énigme leur est posée. Et ainsi de suite jusqu'au dénouement du jeu. Ces énigmes peuvent aller du simple jeu vidéo à terminer sur Internet à la chasse au trésor organisée dans une forêt, en passant par la recherche d'indices au sein d'articles de véritables journaux.

De janvier à juillet 2011, La ludothèque du centre culturel ***Odyssud*** a organisé un *ARG* du nom de *Mahjian* avec le collectif d'artistes ***La Ludopia***. Dans le scénario du jeu, la ville de ***Blagnac***, où se trouve le centre culturel, est menacée par des esprits qui terrorisent les habitants. Ces esprits maléfiques ont été libérés d'un objet mystérieux qui a été ramené au sein de la ludothèque. Mais une organisation secrète, avide d'utiliser le pouvoir de cet objet à de sombres desseins, a cambriolé la ludothèque et dérobé ledit objet. Cela complique d'autant la tâche du couple d'inspecteurs chargés de mener l'enquête, qui sont heureusement aidés par les nombreux joueurs participants à l'aventure. Bien que la dimension ludique de *Majhian* prédomine, un des objectifs fixés aux concepteurs du jeu était de mettre en valeur la commune de ***Blagnac***, commanditaire du projet. Bien qu'il ne s'agisse pas à proprement parler d'un jeu vidéo, nous pouvons donc considérer que *Mahjian* est une forme de Serious Game s'inscrivant dans le marché *« Art & Culture »* combiné à l'intention de *« diffuser un message de communication » (p.28)*. Ainsi, en plus des nombreuses énigmes proposées aux joueurs par Internet, des chasses au trésor dans les parcs de la ville et des animations au sein du centre culturel ont été organisées dans le cadre de ce jeu.

Parmi les nombreux éléments multimédia qui ont du être produits pour animer la fiction de *Mahjian*, se trouve un petit jeu de labyrinthe intégré au sein d'un site Internet rempli d'énigmes. Nous avons utilisé *LudoForge* pour réaliser ce petit jeu très simple. Le joueur doit déplacer un point lumineux avec la souris pour l'amener vers la sortie du labyrinthe. En soi, la création de ce petit jeu reposant uniquement sur cinq *« Règle »* et trois *« Elément »* n'apporte rien de particulier à notre expérimentation, si ce n'est une validation supplémentaire de *LudoForge* sur le terrain. Cependant, pour la réalisation de ce projet nous devions travailler avec les créatifs du collectif ***La Ludopia***, qui ont conçu le jeu *Mahjian*. Afin de pouvoir travailler facilement avec ces artistes pas forcément familiers avec le jeu vidéo, nous avons commencé à développer un premier *« éditeur visuel »* permettant de créer les *« Règle »* de *LudoForge* sans passer par un langage de programmation. Lorsque le jeu tourne, il est possible d'appuyer sur un bouton pour mettre le jeu en pause et afficher la liste de toutes ses *« Règle »*. Les différents paramètres des « conditions » et « actions » des *« Règle »* peuvent alors être modifiés, de même que des « conditions », « actions » et *« Règle »* peuvent être ajoutées ou supprimées. En fermant cet éditeur, le jeu continue son cours en utilisant les *« Règle »* ainsi modifiées.

Pour la réalisation du jeu de labyrinthe, nous avons utilisé cette possibilité de « modification à la volée » *(p.264)* afin de tester rapidement et simplement plusieurs manières pour le joueur de déplacer le point lumineux dans le labyrinthe. Nous avions à côté de nous un des créatifs de ***La Ludopia*** qui pouvait immédiatement tester les *« Règle »* que nous avions modifiées, et voir si cela correspondait à ce qu'il souhaitait proposer aux joueurs. Loin d'être finalisé, cette ébauche *« d'éditeur visuel »* ne permet pas encore d'enregistrer durablement les modifications *(elles disparaissent lorsque le jeu est quitté)*. Une fois que nous avions trouvé des *« Règle »* satisfaisant les attentes des créateurs de *Mahjian*, il nous fallait alors les réécrire en utilisant le langage de programmation de *Flash*. Mais rien n'empêche techniquement la sauvegarde des instances de *« Règle »* ainsi modifiées avec cet *« éditeur visuel »*. Nous continuerons donc à développer cet outil dans les évolutions futures du projet, de manière à réaliser une « interface auteur » accompagnant le « moteur d'interprétation » déjà réalisé.



104. Le jeu vidéo de labyrinthe réalisé avec *LudoForge* pour *Mahjian* et « l'éditeur visuel » associé

2.2.3 BILAN DE CETTE EXPERIMENTATION ET EVOLUTIONS FUTURES

Si la thèse et le *CIFRE* qui l'accompagne arrivent à leur fin, le projet *LudoForge* est loin d'être achevé. Tel qu'exposé dans ce chapitre, nous avons utilisé le contexte de la thèse pour définir tout d'abord un cahier des charges des fonctionnalités attendues pour un outil technique utilisable dans le cadre de nos cours. Au-delà des premières expérimentations du projet *Gam.B.A.S.*, le projet *LudoForge* a permis d'aboutir à la réalisation d'un premier prototype de « moteur d'interprétation » utilisable pour la création de jeu vidéo. Nous avons utilisé ce moteur pour créer quatre jeux vidéo présentés dans ce chapitre. Si les résultats d'utilisation de ce premier prototype sont encourageants, il reste encore de nombreuses étapes à franchir pour aboutir à un outil technique répondant pleinement au cahier des charges. A ce stade du projet, nous voyons trois pistes principales pour les évolutions futures de *LudoForge*. Elles concernent toutes la création d'une « interface auteur » venant compléter le « moteur d'interprétation » existant *(p.257)* :

2.2.3.1 La création d'un véritable « éditeur visuel »

L'étape la plus importante de la suite de ce projet est la création d'un *« éditeur visuel »* qui soit à même de permettre la création des « règles de jeu », et donc du *« Compute »*, sans que le concepteur du jeu ait besoin d'utiliser un quelconque langage de programmation *(p.260)*. L'architecture du « moteur d'interprétation », qui découpe manière très précise les *« Elément »* et les *« Règle »* d'une manière proche de la pensée d'un Game Designer a d'ailleurs été conçue en ce sens *(p.258)*. Maintenant que toutes les classes d'objets nécessaires à la réalisation d'un jeu vidéo existent, il suffit de créer un outil permettant visuellement de créer, modifier et supprimer des instances de ces différentes classes. Nous avons d'ailleurs déjà commencé à réaliser un *« éditeur visuel »* basique allant dans ce sens *(p.291)*. Il faut maintenant rajouter à cet éditeur la possibilité de sauvegarder directement les règles dans un format qui sera ensuite lisible par le « moteur d'interprétation ». Nous nous appuierons vraisemblablement sur du *XML*.

C'est également au sein de cet *« éditeur visuel »* que nous pourrons réintégrer la notion de « briques », de manière à obtenir un outil technique alliant la simplicité d'accès des usines à jeux « spécialisées » avec la puissance créative de celles qui sont « généralistes » *(p.261)*. Concrètement, nous voulons donner au concepteur la possibilité de créer ses propres « briques », en associant différentes instances de *« Règle »* qu'il liera éventuellement à des paramètres commun à toutes les *« Règle »*. Le tout en utilisant uniquement *« l'éditeur visuel »*. Par exemple, supposons qu'un utilisateur crée un jeu dans lequel le joueur peut déplacer un avatar dans quatre directions. Pour cela, il créera quatre *« Règle »*, de la forme *« quand le joueur appuie sur la touche X, déplacer l'élément Y dans la direction Z »*. Mais au lieu de créer quatre *« Règle »* isolées, l'utilisateur pourra « grouper » ces *« Règle »* dans une brique. Il pourra alors définir des paramètres associés à la brique, ces paramètres étant ensuite

utilisables à l'intérieur des *« Règle »* regroupées dans la brique. Pour continuer avec notre exemple, si l'utilisateur crée une brique « mouvement » contenant ses quatre *« Règle »*, il pourra définir une propriété « cible » au niveau de la brique. Il utilisera ensuite cette propriété directement comme paramètre pour les « conditions » et « actions » des *« Règle »* contenues dans la brique. En associant ses quatre *« Règle »* à cette propriété « cible », il pourra donc définir simplement l'instance d'*« Elément »* visée par les quatre règles. De plus, l'utilisateur pourra également réutiliser cette brique à loisir, voire même la partager sur la plateforme « Jeu 2.0 » qui sera associée à l'outil. Nous donnerons ainsi aux concepteurs la possibilité d'utiliser des « briques » en la forme de groupes de *« Règle »* associés à des paramètres de configuration de « haut niveau ». Mais cette simplicité de création des *« Règle »* ne limitera en rien la créativité des utilisateurs, car ils resteront libres de modifier le contenu d'une brique comme ils le souhaitent, toujours en utilisant *« l'éditeur visuel »*. Les modalités exactes d'implémentation d'une telle fonctionnalité restent à définir, mais nous penchons pour l'instant vers la création d'une classe *« Brique »*, spécifique à *« l'éditeur visuel »*, qui permettrait de gérer ces groupes de *« Règle »* et leurs propriétés. Lors de la sauvegarde des *« Règle »*, la notion de *« Brique »* ne serait pas conservée de manière à proposer uniquement une liste linéaire de *« Règle »* au « moteur d'interprétation » pour éviter de le surcharger. Mais peut-être sera t'il au final plus pertinent d'implémenter directement une classe *« Brique »* dans le moteur lui-même. Cet aspect là sera donc vraisemblablement sujet à plusieurs expérimentations avant de trouver l'approche la plus pertinente.

Enfin, nous souhaitons si possible essayer de réaliser un *« éditeur visuel »* qui reste agréable à manipuler, voire même qui possède un aspect amusant, dans l'espoir de pouvoir proposer cet outil à un public jeune *(p.266)*. La réflexion reste encore ouverte sur ce point, même si nous trouvons que la création de *« Règle »* s'apparentant à la manipulation de pièces de puzzle proposée par *Scratch (p.187)* est un bon compromis entre efficacité et amusement.

2.2.3.2 L'intégration d'un outil théorique

Fort d'un *« éditeur visuel »* permettant de créer des *« Règle »*, nous souhaiterions revenir sur l'intégration d'un outil théorique au sein de *LudoForge (p.263)*. Certes, *« l'éditeur visuel »* permettra d'une certaine manière d'utiliser la notion de *« brique de GamePlay » (p.267)*. Mais au-delà de cet outil facilitant la manipulation et la création des « règles du jeu », nous souhaiterons intégrer un outil théorique destiné à guider les étapes *« Définir »* et *« Imaginer »* du *modèle générique DICE (p.103)* de conception d'un Serious Game. Nous avons déjà évoqué comme exemples le langage visuel *WEEV* intégré à *<e-Adventure> (p.222)*, la collection de *« widgets »* imaginés par *Marfisi-Schottman & al. (p.91)* ou tout simplement *« l'écran à idées »* de *Adventure Author (p.225)*.

La réflexion reste ouverte quant au choix de l'outil théorique que nous souhaiterions intégrer à notre outil technique. Une idée qui nous semble intéressante à explorer serait d'utiliser l'alchimie de ***Cook*** *(p.73)* en la modifiant pour prendre en compte l'étape *« Définir »* spécifique à la conception de Serious Games. Ce modèle théorique permet de formaliser schématiquement les différentes « compétences » qu'un joueur doit apprendre pour gagner à un jeu donné. Il permet ensuite de relier ces différentes « compétences » à l'énoncé des différentes « règles du jeu ». Nous pouvons alors imaginer que les « compétences » soient plutôt utilisées pour formaliser le « contenu sérieux » devant être diffusé par le Serious Game. Ce « contenu sérieux » serait ainsi relié aux descriptions textuelles des « règles du jeu » à la place des « compétences » de ***Cook***. En l'état, un outil permettant de dessiner de tels schémas serait déjà un moyen d'aider le concepteur dans sa réflexion créative. Mais intégré à *LudoForge*, nous pouvons imaginer relier chacune des descriptions textuelles de « règles du jeu » à une instance de *« Règle »* dont les « conditions » et « actions » auront été paramétrées en accord avec la description. La création de ces *« Règle »* aurait lieu à travers *« l'éditeur visuel »* mentionné précédemment. Cela ne reste qu'une hypothèse, mais nous pensons que le

fait de pouvoir ainsi établir des connexions directes entre la formalisation « textuelle » des « règles du jeu » et leur implémentation dans un formalisme permettant la création d'un jeu vidéo fonctionnel serait particulièrement appréciable pour les concepteurs. Une expérimentation logicielle s'impose alors pour tenter d'implémenter une telle fonctionnalité une fois que *« l'éditeur visuel »* sera terminé.

2.2.3.3 L'intégration à une plateforme Web 2.0

Enfin, nous souhaiterions également intégrer *LudoForge* au sein d'une plateforme de type *« Web 2.0 »*, de manière à proposer une véritable usine à jeux de type « Jeu 2.0 » *(p.264)*. Nous avons de nombreuses étapes à franchir avant d'entamer la réflexion relative à ce point du cahier des charges, mais nous trouvons la solution mise en place par *Sims Carnival* *(p.202)* particulièrement intéressante. Une fois un jeu créé avec un des trois outils proposé, il est automatiquement partagé sur ce site Internet de manière à ce que tout le monde puisse y jouer. Si un joueur trouve un jeu intéressant au point de vouloir en créer un *« mod »* *(p.51)*, il lui suffit de cliquer sur un bouton pour ouvrir le fichier source de ce jeu dans l'outil technique idoine. Il lui est alors possible d'en créer simplement une variante, qu'il pourra ensuite repartager sur le site. Il faut bien évidemment conserver les « filiations » entre les différents jeux ainsi créés, mais cette approche semble particulièrement pertinente pour s'adresser à un public novice en matière de création vidéoludique. Au lieu de partir d'une base vierge, ils peuvent ainsi étudier les créations d'autres concepteurs, tout en ayant la possibilité de partager à leur tour les jeux qu'ils créent. Les concepteurs souhaitant partir d'une base vierge pourront également le faire s'ils le souhaitent. Mais rien ne les empêchera alors de reprendre uniquement des images et sons provenant d'autres jeux, fonctionnalité régulièrement demandée par nos étudiants *(p.253)*.

La création d'une telle plateforme permettra également de réfléchir à la dimension « travail collaboratif » *(p.265)*. De nombreuses bonnes idées existent sur ce point, à l'image du module de chat intégré à *StencylWorks* pour permettre les échanges au sein d'une équipe de concepteurs. Enfin, la mise à disposition d'un outil technique selon les principes du « Jeu 2.0 » invite à réfléchir à la mise en place d'un tutorial de prise en main de l'outil sous forme de jeu *(p.265)*. Si les exemples sont plus rares sur ce point, nous pouvons déjà nous tourner du côté de *LittleBigPlanet* et *Gamestar Mechanic* qui proposent de véritables jeux vidéo permettant de découvrir tous les « éléments » utilisables dans leur éditeur de niveau. Au final, si la création d'une plateforme de type « Jeu 2.0 » est loin d'être triviale, elle ne pourra vraisemblablement intervenir qu'une fois « l'interface auteur » de *LudoForge* finalisée.

3 SYNTHESE : PREMIERS PAS VERS UN NOUVEL OUTIL TECHNIQUE

Comme nous l'avons évoqué à maintes reprises dans cette thèse, chaque contexte de création de Serious Games appelle à l'utilisation d'outils techniques différents. Après avoir longuement étudié des outils aux applications diverses, nous avons décidé de nous focaliser sur un contexte d'application bien précis : les cours que nous dispensons autour du *« Serious Game Design »* *(p.240)*. En synthèse de nos réflexions sur les approches de facilitation de la création du Serious Games étudiés dans les trois premières parties de cette thèse, nous avons défini le cahier des charges d'un outil adapté à ce contexte *(p.257)*. Au-delà de la définition de fonctionnalités idéales, leur implémentation au sein d'un outil technique fonctionnel est un travail de très longue haleine.

Nous avons commencé une première tentative de réalisation d'un outil technique adapté à ce contexte à travers le projet *Gam.B.A.S.* *(p.267)*. Si les trois prototypes réalisés dans le cadre de ce projet n'ont pas permis de réaliser un outil répondant à nos attentes, ces expérimentations logicielles ont été riches d'enseignements *(p.284)*. Les résultats du projet *Gam.B.A.S.* nous ont permis d'explorer une direction différente à travers une seconde tentative de réalisation d'un outil répondant à notre cahier des charges : *LudoForge* *(p.284)*. Face à l'ampleur de ce nouveau projet, nous avons décidé de fixer un point d'étape intermédiaire à atteindre pour la fin de la thèse : la réalisation d'un « moteur d'interprétation ». Les premiers résultats de l'utilisation de ce « moteur d'interprétation » pour la création de jeux vidéo et Serious Games semblent très encourageants *(p.289)*. Le projet *LudoForge* semble donc promis à de nombreuses évolutions futures *(p.293)*, qui s'accompagneront vraisemblablement de plusieurs expérimentations de terrain auprès de publics d'apprenants variés pour affiner notre approche de l'utilisation du *« Serious Game Design »* comme méthode pédagogique…

BILAN DE LA PARTIE IV : UNE APPLICATION PEDAGOGIQUE DU SERIOUS GAME DESIGN

Appliquant les approches de facilitation de la création vidéoludique identifiées précédemment à un contexte précis afin de les évaluer, cette quatrième et dernière partie de la thèse semble apporter des éléments de réponse à notre problématique *(p.14)* : *Quelles sont les approches permettant de faciliter la création de Serious Games ?*

Dans les premières parties de cette thèse *(p.152 et p.231)*, nous avons observé que **la principale approche permettant de faciliter le « Serious Game Design » consiste à proposer des outils adaptés**. Cette approche « d'ouverture » de la création de Serious Games à de nombreux utilisateurs présente des intérêts dans plusieurs domaines. Tout d'abord, au sein de l'industrie du Serious Game, où elle permet aux concepteurs professionnels d'utiliser des outils de conception et de prototypage entraînant un gain de productivité. Elle représente également un moyen d'impliquer plus facilement les « experts » ou les commanditaires des projets, qui ne sont pas forcément familiers avec la création de Serious Games *(p.214)*. Dans un autre registre, cette approche présente un intérêt certain pour le secteur de l'éducation. Par exemple, elle peut permettre à des enseignants de créer des « matériaux pédagogiques » à même de captiver l'attention de leurs élèves *(p.220)*. Mais il est également possible d'impliquer directement ces élèves dans une pédagogie constructionniste reposant sur la création de Serious Games comme support d'apprentissage *(p.235)*.

De prime abord, le recours à la création de Serious Games comme support d'activité pédagogique apparaît particulièrement intéressante. En impliquant l'apprenant dans la réalisation d'un projet, cette approche permet de le sensibiliser aussi bien aux problématiques de conception d'un Serious Game qu'au « contenu sérieux » que son jeu devra transmettre. Mais cette application pédagogique du *« Serious Game Design »* constitue également une des approches permettant de le faciliter. En effet, la présence d'un pédagogue pour accompagner et guider les apprenants dans leur processus de création de Serious Games participe activement à l'adaptation du *« Serious Game Design »* aux compétences des utilisateurs. Si le recours à un outil technique adapté est toujours nécessaire, l'aspect théorique de la création de Serious Games reposera par contre plus facilement sur le pédagogue. Comme nous l'avons expérimenté durant nos propres cours, un « médiateur humain » sera bien plus efficace qu'un « outil théorique » pour accompagner la réflexion créative des apprenants *(p.241)*. **La mise à disposition d'outils adaptés n'est donc pas la seule approche permettant de faciliter le « Serious Game Design » : le fait d'avoir un référent spécialiste du sujet pour accompagner les concepteurs représente également une approche très pertinente.**

Au-delà du rôle primordial des référents humains, **le contexte en lui-même peut parfois jouer un rôle pour l'accessibilité de la création de Serious Games**. Pour reprendre notre exemple précédent, en utilisant le *« Serious Game Design »* comme méthode pédagogique, la création de Serious Games dépasse le simple cadre de la création vidéoludique *(p.245)*. Elle devient une activité pratiquée dans un contexte pédagogique qui vise une finalité éducative. Le *« Serious Game Design »* est alors potentiellement attractif pour un public qui n'est pas forcément intéressé par la seule création vidéoludique, mais qui reste sensible à l'apprentissage selon des modalités constructionnistes. Par exemple, si peu d'élèves de

collèges s'adonnent à la création de jeux vidéo pendant leurs temps de loisirs, nombre d'entre eux seront sûrement motivés par la possibilité d'apprendre à l'école à travers la création de Serious Games *(p.206)*.

Cependant, malgré l'importance du contexte d'application du *« Serious Game Design »*, le principal pilier de toutes ces applications reste bel et bien l'outil technique qui sera proposé aux concepteurs. Celui-ci doit être adapté aussi bien aux désirs créatifs de l'utilisateur qu'à son niveau de connaissance technique, tout en prenant en compte les objectifs industriels *(productivité)* ou éducatifs *(apprentissage)* du contexte d'utilisation *(p.255)*. **Nous retrouvons alors ici l'importance de disposer d'une large sélection d'outils techniques, assortie d'un moyen de les classifier**, tel que nous l'avons proposé dans la partie précédente *(p.231)*. Malgré la richesse des logiciels disponibles, il peut arriver qu'aucun outil technique ne soit pleinement adapté à un contexte donné. Le retour d'expérience de nos étudiants sur l'utilisation d'une sélection des outils analysés dans cette thèse illustre cette situation. Si les Serious Games qu'ils ont été en mesure de produire montrent la pertinence des outils existants *(p.243)*, nous avons cependant identifié de nombreuses petites limites dont ces outils souffrent dans ce contexte d'utilisation précis *(p.248)*. En réponse à ces limites, nous avons formalisé dans un « cahier des charges » la description d'un « outil idéal » pour le contexte de nos cours *(p.257)*. Ce cahier des charges propose des points s'inspirant d'outils existants, comme par exemple la construction d'une architecture logicielle proche du mode de pensée d'un Game Designer *(p.258)*. Il propose également des fonctionnalités peu répandues, comme l'intégration d'un outil théorique inspiré de ceux du *« Game Design »* au sein de l'outil technique *(p.263)*. **Ce « cahier des charges » synthétise donc l'ensemble de nos réflexions sur les approches, propres aux outils techniques, permettant de faciliter le « Serious Game Design »**. Ce faisant, il pose une nouvelle problématique de recherche : *est-il possible de réaliser un outil répondant à cette description d'un « outil idéal » ?*

La recherche de la réponse à cette problématique augure nos futurs travaux dans le domaine. Nous avons déjà commencé à mettre en place des expérimentations logicielles pour tenter de réaliser un outil répondant à ce cahier des charges. Tout d'abord, le projet *Gam.B.A.S. (p.267)*, qui s'appuie sur l'outil théorique des *« briques de GamePlay » (p.267)*, vise à proposer un outil technique facilitant la création de Serious Games. Si cette première expérimentation n'a pas abouti, elle a cependant été riche d'enseignements sur les nombreuses difficultés que représente la réalisation d'un outil répondant à notre cahier des charges *(p.284)*. Les résultats de cette première expérimentation ont permis d'explorer une direction différente avec le projet *LudoForge (p.284)*. Bien que ce projet soit toujours en cours, ses résultats à mi-parcours semblent particulièrement encourageants : le « moteur d'interprétation » sur lequel reposera ce futur outil technique a déjà permis la réalisation d'un des Serious Games présenté dans cette thèse *(p.290)*. Nous espérons donc avoir l'opportunité de continuer le projet *LudoForge*, selon les directions qui ont été définies *(p.293)*, au-delà du cadre de cette thèse et du *CIFRE* qui l'accompagne.

CONCLUSION

S'inscrivant dans le contexte des « Serious Games », le travail de recherche présenté dans cette thèse s'articule autour de la problématique suivante : **Quelles sont les approches permettant de faciliter la création de Serious Games ?**

1 DEMARCHE

Afin de tenter d'apporter une réponse à cette problématique, nous proposons d'explorer les approches mises en place par le secteur du jeu vidéo de divertissement pour permettre aux créateurs amateurs de concevoir leurs propres jeux. Nous avons commencé par définir les deux domaines dans lesquels s'inscrit notre travail : les *« Serious Games »* *(p.18)* et le *« Game Design »* *(p.42)*.

Du côté des *« Serious Games »*, nous avons vu qu'il était possible de les définir comme **« tout jeu explicitement conçu pour une finalité dépassant le cadre du divertissement »**. Cette définition générale peut s'appliquer à tout type de jeu, mais la « vague actuelle » de Serious Games, qui débute en 2002, se caractérise par la prédominance du support vidéoludique *(p.18)*. Nous avons donc choisi de nous intéresser uniquement aux Serious Games basés sur le jeu vidéo *(p.55)*. En ce qui concerne le *« Game Design »*, nous avons pu identifier trois définitions pour ce terme :

- Le processus global de création d'un jeu *(p.42)*.
- La phase de conception d'un jeu, distincte de la phase de fabrication *(p.44)*.
- Une partie de la phase de conception : la création des mécanismes et de l'univers de jeu. Cette partie se distingue de celle du *« Level Design »*, qui correspond à la construction des niveaux du jeu *(p.45)*. Elle diffère également de celle du *« Play Design »*, qui correspond à l'étape d'affinage du jeu pour délivrer la meilleure expérience possible au joueur *(p.149)*.

Dans le cadre de cette thèse, nous avons choisi de nous inscrire dans la définition du *« Game Design »* en tant que *« processus global de création d'un jeu »* *(p.54)*. Suite à ce préambule, nous proposons de **définir le « Serious Game Design » comme le « processus global de création d'un Serious Game »** *(p.56)*. Notre étude préliminaire du *« Game Design »* nous a également permis de constater que la création de jeux vidéo implique deux volets : une phase de travail **« théorique »**, qui consiste à imaginer le concept du jeu, et une phase **« technique »** qui consiste à l'implémenter au sein d'un programme logiciel *(p.55)*.

Notre travail de recherche s'articule selon ces deux volets. Nous avons tout d'abord étudié les considérations « théoriques » relatives au *« Serious Game Design »*. Nous avons commencé par identifier les différents « outils théoriques » utilisés par les professionnels des secteurs du « Game Design » *(p.59)* et du « Serious Game » *(p.86)*. Après avoir comparé les méthodologies destinées aux jeux vidéo *(p.65)* avec celles pensées pour les Serious Games *(p.103)*, nous observons que le processus de conception d'un Serious Game se distingue effectivement de celui d'un jeu vidéo. En nous appuyant sur des retours d'expérience liés à la création de Serious Games dans le cadre du *CIFRE*, nous avons identifié de manière plus précise plusieurs spécificités d'ordre théorique liées au *« Serious Game Design »* *(p.106)*. Nous observons cependant qu'**au-delà du processus de conception de Serious Games, il n'existe pour l'instant que très peu d'approches visant à faciliter l'aspect théorique de sa création** *(p.150)*.

En revanche, l'étude des considérations « techniques » du *« Serious Game Design »* nous amène à **identifier de nombreuses approches facilitant la dimension technique de la création de Serious Games**. Nous avons pour cela analysé plusieurs catégories d'outils techniques provenant du champ du *« Game Design »*, comme les *« usines à jeux »* *(p.166)*, le *« Jeu 2.0 »* *(p.200)* et les *« outils de modding »* *(p.163)*, ainsi que quelques outils explicitement dédiés à la création de Serious Games *(p.214)*. Si les outils théoriques de *« Game Design »* et de *« Serious Game Design »* possèdent de nombreuses différences, les outils techniques provenant de ces deux domaines sont en revanche très proches. Le *« Serious Game Design »* peut donc être facilité en s'appuyant sur les nombreux outils techniques destinés à mettre le *« Game Design »* à la portée de novices. La création de Serious Games dispose ainsi d'une vaste base d'outils techniques variés lui permettant d'être potentiellement accessible à un large public *(p.230)*.

Nous avons alors éprouvé cette approche dans un contexte d'application concret : des cours destinés à sensibiliser des étudiants aux problématiques de conception vidéoludique *(p.234)*. Si cette expérimentation nous a permis d'évaluer la pertinence de plusieurs outils précédemment identifiés *(p.248)*, elle nous amène également à constater que la proposition d'outils adaptés à des novices n'est pas la seule approche de facilitation du *« Serious Game Design »*. En effet, le contexte pédagogique de ces cours représente également un facteur d'accessibilité de la création de Serious Games. D'une part, l'enseignant joue un rôle primordial pour accompagner les étudiants dans le processus créatif, leur permettant de réaliser des Serious Games très intéressants *(p.243)*. Mais surtout, le contexte pédagogique dans lequel s'inscrit ainsi le *« Serious Game Design »* semble représenter, aux yeux de certains apprenants, un facteur de motivation supplémentaire pour découvrir cette activité *(p.297)*.

2 CONTRIBUTION

En résumé, pour répondre à notre problématique, nous identifions deux types d'approches permettant de faciliter la création de Serious Games : **la mise à disposition d'outils adaptés** et **la mise en place d'un contexte d'accompagnement**. D'une manière plus précise, voici ce que nous avons pu découvrir sur ces deux approches :

2.1 LA MISE A DISPOSITION D'OUTILS ADAPTES

Les outils qu'il est possible de proposer pour faciliter la création de Serious Games sont de deux types : les **« outils théoriques »** et les **« outils techniques »** *(p.55)*.

2.1.1 LES OUTILS THEORIQUES

Dans le champ du *« Game Design »* nous observons deux types « d'outils théoriques » : les méthodologies de conception *(p.60)* et les outils d'aide à la réflexion créative *(p.68)*. Dans le cas des outils d'aide à la réflexion créative, ils sont destinés à guider le concepteur dans l'invention d'un concept de jeu. Pour cela, ils se focalisent sur trois aspects :

- *La structure du jeu (p.68)*.
- *Le profil du joueur (p.71)*.
- *La relation entre le joueur et le jeu (p.73)*.

Bien qu'à ce jour ces outils n'aient pas été directement utilisés pour les Serious Games, ils semblent pourtant adaptés à leur création : les aspects sur lesquels ils portent sont d'ordre « formel », et donc communs au jeu vidéo de divertissement et au Serious Game *(p.56)*. Cependant, que ce soit dans un contexte professionnel *(p.87)* ou pédagogique *(p.248)*, nous

avons pu constater que ces outils étaient relativement complexes à utiliser pour des novices. A défaut de pouvoir être appliqués directement, ils sont néanmoins potentiellement utilisables comme source d'inspiration pour la création d'outils théoriques plus simples *(p.294)*.

Du côté des méthodologies de conception, il ne semble pas possible de proposer une formalisation universelle du processus de *« Game Design »* *(p.60)*. Nous souhaitons donc introduire un « modèle générique », qui propose des étapes générales à partir desquelles chaque concepteur peut définir une « série d'étapes » qui lui est propre. Suite à l'analyse des modèles existants pour la conception de jeux vidéo de divertissement, nous avons proposé le *modèle générique ICE*, composé de trois étapes : *« Imaginer »*, *« Créer »* et *« Evaluer »* *(p.65)*. Nous avons également identifié d'autres méthodologies formalisant cette fois-ci le processus de conception d'un Serious Game *(p.91)*. En les analysant selon une démarche similaire, nous proposons le *modèle générique DICE*, composé de quatre étapes : *« Définir »*, *« Imaginer »*, *« Créer »* et *« Evaluer »* *(p.103)*. En comparant ces deux modèles génériques, nous constatons que **le processus de conception de Serious Games se distingue de celui du jeu vidéo par la présence d'une étape supplémentaire** : la définition d'un contenu sérieux à transmettre. Si la phase *« Définir »* ne possède pas d'outils théoriques dédiés, nous avons rencontré plusieurs exemples de *« devis de conception »* formalisant le travail de cette étape, ainsi qu'une proposition d'utilisation d'outils provenant d'autres domaines comme l'analyse marketing ou la stratégie industrielle *(p.94)*.

Fort de cette analyse comparée des étapes du processus de conception d'un jeu vidéo de divertissement et d'un Serious Game, nous avons décidé de poursuivre notre analyse des spécificités d'ordre théorique du *« Serious Game Design »* *(p.106)*. Pour cela, nous avons étudié de manière détaillée la création de Serious Games réalisés dans le cadre du *CIFRE* accompagnant cette thèse *(p.115)*. Au-delà de la présence de l'étape *« Définir »*, ce retour d'expérience nous a permis d'**identifier d'autres spécificités de la création de Serious Games relatives aux trois étapes du processus de conception qu'il partage avec le jeu vidéo de divertissement** :

2.1.1.1 Spécificités de l'étape « Imaginer »

Lors de cette étape, la conception de Serious Game se distingue principalement de celle de jeu vidéo par le fait d'avoir à associer les dimensions « sérieuse » et « ludique » *(p.106)*. Nous avons identifié deux approches principales pour associer ces deux dimensions :

- *L'approche « extrinsèque »*. Inspirée par les théories béhavioristes, cette approche vise à utiliser les deux dimensions de manière séparée au sein du Serious Game.

- *L'approche « intrinsèque »*. Associée aux théories cognitivistes, cette approche consiste à mélanger les deux dimensions de manière à ce qu'il ne soit plus possible de les distinguer.

Le fait de s'appuyer sur l'une ou l'autre de ces deux approches change considérablement le travail de conception. En accord avec les tendances de la « vague actuelle » de Serious Games, nous avons particulièrement exploré l'approche « intrinsèque » *(p.108)*. Nous constatons alors qu'il semble exister deux principaux « vecteurs » permettant de diffuser un « contenu sérieux » pour un Serious Game de type « intrinsèque » *(p.142)* :

- *Par le « contenu du jeu »*. Le contenu sérieux est présenté de manière explicite à l'intérieur du jeu. Le principe de jeu est alors pensé pour que le joueur s'y intéresse.

- *Par les « règles du jeu »*. Le contenu sérieux est intégré au sein des mécanismes de jeu. Le joueur doit alors analyser les règles du jeu afin d'identifier le contenu sérieux qu'il véhicule.

Au-delà de ces différences fondamentales, l'étape *« Imaginer »* de la création d'un Serious Game se rattache à celle d'un jeu vidéo sur certains points. Par exemple, le choix entre les approches *« gameplay expérimental »* et *« jeu de genre »* *(p.140)* se retrouve également pour la création d'un jeu vidéo de divertissement. Dans un autre registre, la mise en place de systèmes de « tutoriaux » expliquant le fonctionnement du jeu n'est pas spécifique au Serious Game *(p.147)*. Bien que cette approche ne soit pas aussi importante pour l'ensemble des jeux vidéo de divertissement, elle est primordiale pour un de ses courants : le *« Casual Game »*. De par la nécessité d'expliquer de manière exhaustive les codes de la culture vidéoludique afin de pouvoir s'adresser à des joueurs novices, la conception de Serious Games rejoint donc celle de *« Casual Games »* sur certains aspects.

2.1.1.2 Spécificités de l'étape « Créer »

La création de prototypes, que ce soit pour un projet de Serious Game ou de jeu vidéo de divertissement, s'appuie principalement sur des outils techniques plutôt que théoriques. Nous reviendrons plus en détail sur les outils techniques dans la section suivante *(p.303)*.

2.1.1.3 Spécificités de l'étape « Evaluer »

Si les jeux de divertissement doivent être évalués tout comme les Serious Games, ils se différencient par les critères qui seront utilisés pour cette évaluation. Le principal critère d'évaluation d'un jeu de divertissement est la qualité du « fun » qu'il procure au joueur. Si les Serious Games se doivent d'être amusants, ils doivent également être efficaces dans la transmission de leur contenu sérieux. Les critères d'évaluation d'un Serious Game porteront donc à la fois sur ses dimensions « sérieuse » et « ludique ».

Plusieurs méthodes existent pour évaluer un Serious Game. Dans le cas d'un jeu basé sur une approche « intrinsèque », nous avons par exemple pu observer l'intérêt des méthodes de « suivi du joueur » pour évaluer l'impact d'un Serious Game *(p.145)*. Si de telles méthodes peuvent également être mobilisées par le jeu vidéo de divertissement, elles semblent être plus volontairement associées au Serious Game. Nous avons d'ailleurs nous-mêmes expérimentés cette méthode dans deux Serious Games réalisés dans le cadre du *CIFRE*, grâce à des outils de mesure génériques encore peu utilisés pour les Serious Games *(p.147)*. Conjuguée à une analyse qualitative *(p.129)*, l'analyse quantitative des données obtenues par l'intégration de fonctionnalités de suivi du joueur au sein de ces deux Serious Games s'est avérée très pertinente pour l'évaluation de l'impact de nos Serious Games *(p.134)*.

2.1.1.4 Approches de facilitation des spécificités du Serious Game Design

Bien que ces spécificités représentent des aspects fondamentaux de la dimension « théorique » du processus de conception d'un Serious Game, elles ne semblent pas à ce jour être véritablement facilitées par des outils théoriques. Comme nous avons pu le constater lors d'expérimentations en contexte industriel *(p.87)* et pédagogique *(p.248)*, les outils d'aide à la réflexion créative que nous avons identifiés sont relativement complexes. Cela les rend difficilement utilisables par des novices en matière de création vidéoludique. Seules les méthodologies de conception semble facilement accessibles à des novices.

Notre principale contribution aux approches de facilitation de l'aspect théorique de la création de Serious Games est donc la formalisation de notre méthodologie de conception personnelle *(p.150)*. Cette méthodologie de conception synthétise les différentes spécificités du Serious Game Design que nous avons pu identifier lors de la réalisation des quatre projets Serious Game qui s'inscrivent dans le *CIFRE* accompagnant cette thèse *(p.86, p.115 et p.140)*. En synthétisant ainsi les spécificités de la création de Serious Games, il sera plus facile pour un médiateur humain, par exemple un enseignant, de les assimiler pour les transmettre à des novices grâce à la mise en place d'un contexte d'accompagnement adéquat *(p.305)*.

2.1.2 LES OUTILS TECHNIQUES

La situation est très différente du côté des outils techniques. D'une part, le champ du *« Game Design »* dispose de très nombreux outils permettant à des novices de créer des jeux vidéo *(p.163, p.166 et p.200)*. Mais surtout, **ces nombreux outils du « Game Design » sont tout à fait utilisables pour la création de Serious Games**. Cette conclusion s'appuie sur les nombreuses expérimentations en la matière menées par d'autres chercheurs ou pédagogues *(p.231)*, ainsi que sur notre propre retour d'expérience suite la création d'un « Serious Game » par un de nos stagiaires grâce à l'usine à jeux *The Games Factory 2 (p.195)*. Il existe également quelques outils techniques explicitement destinés à la création de Serious Games *(p.214)*. Nous observons alors qu'ils offrent globalement les mêmes fonctionnalités que ceux du *« Game Design » (p.228)*. Lors de notre *CIFRE*, nous avons même pu constater que les « usines à jeux » pouvaient être une source d'inspiration très riche pour l'amélioration d'un outil de création de Serious Game *(p.215)*. Notons tout de même que ces outils techniques de *« Serious Game Design »* proposent quelques petites fonctionnalités absentes de ceux du *« Game Design »*. Par exemple, tous les logiciels que nous avons étudiés offrent en standard des méthodes d'implémentation de fonctionnalités de « suivi de joueur » *(p.145)*. Lors de la réalisation de Serious Games dans le cadre du *CIFRE*, nous avons d'ailleurs expérimenté une technique de suivi des joueurs à partir « d'outils de mesure génériques », méthode fort intéressante mais pourtant absente des outils que nous avons pu étudier *(p.134)*. Au-delà de ces différences, tous ces outils techniques proposent donc des approches à même de faciliter la création de Serious Games. Lors de l'analyse d'un corpus de 400 outils techniques, **nous avons identifié les approches de facilitation suivantes** *(p.230)* :

- *La limitation du niveau technologique des jeux créables*. La première approche de facilitation de la création vidéoludique consiste tout simplement à limiter la complexité technologique des jeux qui sont réalisés. Cela impacte notamment le type de rendu graphique utilisé *(p.194)*. Par exemple, nous avons vu des outils spécialisés dans la création de jeux vidéo basés sur un rendu textuel *(p.166)*. La création de la partie graphique de ce type de jeu est bien plus aisée que la création de graphismes 2D, elle-même plus simple que la modélisation de graphismes 3D requise pour la création d'un jeu vidéo en 3D temps réel.

- *La limitation des composantes créables*. Avec le *modèle ISICO (p.159)*, nous avons mis en évidence de manière théorique l'existence de quatre « parties » qu'il faut créer lors de la réalisation d'un jeu vidéo :
  **« Etat Initial »* : l'état initial du système à état variable qu'est le jeu.
  **« Input »* : les moyens proposés au joueur pour envoyer des informations au système.
  **« Compute »* : les mécanismes internes qui permettent au système de changer d'état.
  **« Output »* : la manière dont le système communique son état courant au joueur.
  Nous constatons que, pour faciliter la création d'un jeu vidéo, certains outils ne permettent pas à l'utilisateur de créer lui-même ces quatre composantes. Par exemple, un outil comme *Cartoon Network Game Creator (p.210)* ne permet de créer que l'*« Etat Initial »* du jeu, à travers la construction de niveaux. Les trois autres « parties » du jeu sont générées automatiquement par l'outil, simplifiant ainsi grandement la création de jeux. En contrepartie de sa grande facilité d'accès, ce type d'approche limite sévèrement la créativité des concepteurs.

- *Choix d'un outil spécialisé ou généraliste*. En se spécialisant dans un genre de jeu donné, les outils techniques peuvent proposer de nombreuses parties « préconstruites » qui réduisent considérablement le temps de réalisation d'un jeu vidéo *(p.176)*. Mais cette approche peut parfois limiter la créativité des utilisateurs, et n'est de toute façon pas envisageable pour des outils généralistes. Ces derniers misent plutôt sur l'accessibilité de leur interface pour simplifier la création vidéoludique *(p.177)*.

- *Le recours à des éditeurs de contenu émergeant ou à des menus de choix prédéfinis.* Il s'agit tout simplement des deux grands types « d'outils » proposés par les logiciels de création vidéoludique *(p.160)*. Les menus de choix prédéfinis sont plus simples d'utilisation, mais ils limitent la créativité aux bornes de l'imagination du concepteur de l'outil *(p.184)*. À l'inverse, les éditeurs émergeants permettent de créer du contenu qui n'aura pas forcément été anticipé par le concepteur de l'outil, au prix d'une manipulation un peu plus complexe *(p.168)*. Chaque outil technique utilise une combinaison des deux approches qui lui est propre. Ce choix influe directement sur l'équilibre entre la liberté de création et la simplicité d'utilisation de chaque logiciel.

- *Choix d'un éditeur visuel ou d'un langage de programmation pour les règles de jeu*. Les *« éditeurs visuels »* permettent de créer des programmes informatiques sans avoir recours à un langage de programmation. Ils sont donc bénéfiques pour l'accessibilité de l'outil, en particulier auprès d'un public novice *(p.180)*. D'un autre côté, les outils basés sur un langage de programmation sont un bon moyen d'apprentissage de l'informatique *(p.187)*. L'emploi d'un langage de programmation commun peut même constituer un tremplin vers le monde professionnel du jeu vidéo *(p.186)*.

En jouant sur ces différentes approches, les outils que nous avons analysés font preuve d'une grande variété en terme d'accessibilité et de possibilités de création. Si cette large variété d'outils techniques est une aubaine pour faciliter le *« Serious Game Design »* auprès d'une grande diversité de publics, une telle profusion pose néanmoins la question de la pertinence du choix. En effet, comme d'autres travaux l'ont déjà démontré *(p.164)*, **le choix d'un outil technique adapté conditionne grandement la capacité des concepteurs à créer un jeu vidéo ou un Serious Game**. Par exemple, lors de nos évaluations d'outils techniques en contexte pédagogique *(p.240)*, nous avons pu remarquer que la sélection d'un tel logiciel doit répondre à de nombreuses contraintes :

- L'âge et le niveau de compétence technique des apprenants : *maîtrise préalable d'un outil ? Connaissance d'un langage de programmation ? Âge des apprenants pour la compréhension de certains concepts ?…*
- La durée de l'activité pédagogique : *prévoir des outils simples d'accès si les apprenants doivent les découvrir durant un module de cours de faible durée…*
- Le matériel à disposition de l'enseignant : *nombre et puissance des ordinateurs à disposition, modalité d'accès à Internet, possibilité d'installer des logiciels sur les machines de l'établissement…*
- Les fonctionnalités de l'outil selon les projets envisagés : *les apprenants vont-il faire des jeux en 2D ou en 3D ? Ont-ils besoin de créer tous les aspects de leur jeu ? Se concentrent-ils sur un genre vidéoludique précis ou non ?…*

Que ce soit pour un contexte pédagogique, industriel ou tout autre type de contexte, de nombreuses questions analogues à celles-ci se posent pour choisir un outil technique adapté. Afin de faciliter le choix d'un outil technique de *« Serious Game Design »*, nos proposons un système de classification permettant d'apporter une réponse à ces questions : le *modèle ISICO étendu* *(p.160)*. Les critères de ce modèle classificatoire synthétisent les différentes caractéristiques des outils techniques de création vidéoludique, facilitant ainsi le choix d'un outil adapté. Lors de notre travail de recherche, nous avons classifié un corpus de 400 outils techniques avec ce modèle *(p.159)*, ce qui semble pour l'instant indiquer sa pertinence. Cette thèse présente des tableaux classifiant les outils techniques du *« Jeu 2.0 »* *(p.202)* et de *« Serious Game Design »* *(p.228)*. Le détail de l'ensemble du corpus d'outils techniques classifiés avec le *modèle ISICO étendu* est accessible sous forme d'une base de données collaborative en ligne. Nous avons créé ce site Internet afin de faciliter la recherche d'outils techniques, en nous appuyant sur les différents critères de notre modèle classificatoire *(p.162)*. Il est accessible à l'adresse : http://creatools.gameclassification.com

Si la mise à disposition d'outils adaptés est une approche incontournable, il est également possible de mettre en place un contexte d'accompagnement des concepteurs pour leur permettre de créer leurs propres Serious Games. Nous avons pu le constater lors de nos cours auprès d'étudiants *(p.240)*. Si le recours à un outil technique adapté est toujours nécessaire, l'aspect théorique de la création de Serious Games reposera plus facilement sur le pédagogue. Ce dernier peut par exemple palier l'absence d'outils théoriques de *« Serious Game Design »* accessibles aux novices, en sensibilisant ses apprenants aux nombreuses spécificités du Serious Game lors du suivi de leurs projets. Au final, tel que nous l'avons expérimenté durant nos cours, **un « médiateur humain » semble être plus efficace qu'un « outil théorique » pour accompagner la réflexion créative des apprenants** *(p.241)*.

Dans un autre registre, ce retour d'expérience sur la mise en place de cours reposant sur la création de Serious Games nous amène à constater l'énorme potentiel pédagogique de cette approche en regard de la théorie d'apprentissage du « constructionnisme » *(p.235)*. En impliquant l'apprenant dans la réalisation d'un projet créatif, le *« Serious Game Design »* permet de le sensibiliser aussi bien aux problématiques de conception d'un Serious Game qu'au « contenu sérieux » que son jeu devra transmettre. Il est ainsi envisageable d'enseigner de nombreux sujets grâce au *« Serious Game Design »*, en proposant aux apprenants de réaliser un jeu traitant d'un thème sélectionné par le pédagogue. Par exemple, nous avons pu constater qu'en créant des Serious Games, nos étudiants arrivaient à enrichir leurs connaissances sur des sujets de société, parfois même au point d'être en mesure d'en débattre lors de nos cours *(p.245)*.

3 LIMITES

Ces contributions sont bien entendu sujettes à certaines limites. Tout d'abord, d'un point de vue méthodologique, les conclusions présentées ici ne sont pas issues d'une étude exhaustive du domaine. En effet, notre travail repose sur la constitution de différents corpus, qu'il s'agisse d'une sélection de *« Serious Games »* *(p.32)* comme d'outils théoriques *(p.59)* et techniques *(p.159)* de *« Game Design »*. Ensuite, les retours d'expériences proposés dans cette thèse s'appuient sur quelques exemples précis de Serious Games, qui ne sont pas totalement représentatifs des nombreux types de finalités et marchés visés par ces jeux *(p.28)*. Les analyses effectuées et les conclusions proposées s'appuient donc sur un échantillon partiel du champ étudié. Si nous avons tenté de constituer des corpus larges pour limiter les biais potentiels, il faudrait à présent conforter nos conclusions en étudiant des corpus encore plus conséquents.

Cette remarque s'applique également aux contextes d'application du *« Serious Game Design »*. La convention *CIFRE* nous a permis de travailler dans trois contextes « industriels » différents, à savoir une ludothèque, un aménageur urbain et une société de simulation *(p.15)*, auxquels s'ajoute le contexte « pédagogique » de nos cours *(p.240)*. Si nous avons essayé d'illustrer des contextes d'application variés dans cette thèse, ces quatre exemples sont loin de constituer un échantillon représentatif. Plus encore que l'analyse de corpus élargis, l'évaluation de la pertinence des approches permettant de faciliter la création de Serious Games dans d'autres contextes d'application semble donc primordiale.

Dans un autre registre, le travail de recherche présenté dans cette thèse s'est principalement focalisé sur les approches provenant d'un seul domaine, le *« Game Design »*. Peut-être existe-t-il d'autres domaines pouvant permettre d'apporter des réponses à notre problématique ?

Pour autant, nous identifions également des limites liées aux contributions du *« Game Design »* au champ du *« Serious Game »*. Par exemple, malgré la richesse des logiciels disponibles, il peut arriver qu'aucun outil technique ne soit pleinement adapté à un contexte donné. Le retour d'expérience de nos étudiants sur l'utilisation d'une sélection d'outils issus de cette thèse illustre cette situation. Si leurs réalisations montrent la pertinence des outils existants *(p.243)*, nous avons cependant identifié de nombreuses petites limites dans les logiciels existants *(p.248)*. En réponse à ces limites, nous avons formalisé dans un « cahier des charges » la description d'un « outil idéal » pour le contexte de nos cours *(p.257)*. Ce cahier des charges propose des points s'inspirant d'outils existants, tel que la construction d'une architecture logicielle proche du mode de pensée d'un Game Designer *(p.258)*. Il propose également des fonctionnalités peu répandues, comme l'intégration d'un outil théorique inspiré de ceux du *« Game Design »* au sein de l'outil technique *(p.263)*.

4 PERSPECTIVES

Au final, **ce « cahier des charges » synthétise donc l'ensemble de nos réflexions sur les approches, propres aux outils techniques, permettant de faciliter le « Serious Game Design »** *(p.257)*. Ce faisant, il pose une nouvelle problématique de recherche : est-il possible de réaliser un outil répondant à cette description d'un « outil idéal » ?

La recherche de la réponse à cette problématique augure nos futurs travaux dans le domaine. Nous avons déjà commencé à mettre en place des expérimentations logicielles pour tenter de réaliser un outil répondant à ce cahier des charges. Tout d'abord, le projet *Gam.B.A.S. (p.267)*, qui s'appuie sur l'outil théorique des *« briques de GamePlay » (p.267)*, vise à proposer un outil technique simplifiant la création de Serious Games. Si cette première expérimentation n'a pas abouti, elle a cependant été riche d'enseignements sur les nombreuses difficultés que représente la réalisation d'un outil répondant à notre cahier des charges *(p.284)*. Les résultats de cette première expérimentation ont permis d'explorer une direction différente avec le projet *LudoForge (p.284)*. Bien que ce projet soit toujours en cours, ses résultats à mi-parcours semblent particulièrement encourageants : le « moteur d'interprétation » sur lequel reposera le futur outil technique a déjà permis la réalisation d'un des Serious Games présenté dans cette thèse : *Le Jardinier Ecolo (p.290)*. Nous espérons donc avoir l'opportunité de continuer le projet *LudoForge*, selon les directions qui ont été définies *(p.293)*, au-delà du cadre de cette thèse et du *CIFRE* qui l'accompagne.

Nous continuerons également à élargir nos corpus de Serious Games *(p.32)* et d'outils techniques *(p.159)*. En nous appuyant sur des contributions extérieures, nous espérons être un jour en mesure de proposer des analyses sur des corpus approchant l'exhaustivité dans ces deux domaines.

Une autre direction que nous souhaiterions explorer dans nos travaux futurs est l'application du *« Serious Game Design »* à différents contextes. Par exemple, nous aimerions expérimenter l'utilisation de la création de Serious Games comme méthode pédagogique auprès de nouveaux types de publics : *élèves du primaire ou du secondaire, adultes en formation professionnelle, ateliers culturels…* Cela permettrait avant tout de continuer à étudier l'intérêt du *« Serious Game Design »* pour la pédagogie reposant sur la théorie constructionniste *(p.237)*. Mais surtout, cela permettrait de continuer à réfléchir aux approches pouvant mettre la création de Serious Games à la portée de novices. Le fait d'évaluer « sur le terrain » la pertinence des approches existantes au sein de contextes d'applications concrets, comme nous l'avons effectué dans cette thèse, semble être un des meilleurs moyens pour critiquer et améliorer les approches existantes, voire pour en découvrir de nouvelles…

ANNEXES

1 CONSIDERATIONS HISTORIQUES SUR LE GAME DESIGN

Cette section rassemble plusieurs considérations historiques liées au domaine du *« Game Design »* et de ses différentes approches de facilitation *(p.46)*. Nous les nous avons consignées ici afin de ne pas alourdir la démarche de réflexion développée dans cette thèse.

1.1 ORIGINE HISTORIQUE DES GAME DESIGNERS PROFESSIONNELS INDEPENDANTS

Si, jusqu'aux années 80, l'ensemble de l'industrie vidéoludique fonctionnait sur le modèle d'un concepteur impliqué dans la création de la plupart des aspects de son titre, les années 90 marquent un tournant avec l'apparition d'équipes de production de plus en plus importantes. Pour autant, de nombreux concepteurs de jeux ont sciemment fait le choix d'œuvrer en tant qu'indépendants, et de « rester petits » pour garder un contrôle total sur leurs créations. La carrière de ***Jeff Minter*** et de son studio *Llamasoft* en est un parfait exemple. A travers sa société unipersonnelle fondée en 1982, il réalisera de nombreux classiques de l'histoire du jeu vidéo tels que *Girdrunner* ***(1982)***, *Attack of the Mutant Camels* ***(1983)***, *Tempest 2000* ***(1994)*** ou plus récemment *Space Giraffe* ***(2007)***. Si, lors de ses débuts, le parcours de ***Minter*** ne diffère guère des autres professionnels du jeu vidéo, dès les années 90 son mode de production de jeux vidéo en solitaire le distingue clairement du reste de l'industrie (J. Minter & P. Minter, 2009). Pour ces créateurs ayant fait le choix de l'indépendance, se pose la question des moyens de diffusion et de commercialisation de leurs œuvres. Dans les années 70-80, il suffisait de copier un jeu sur disquette et de le vendre de la main à la main ou par courrier postal. Par exemple, c'est ainsi que ***Richard Garriot***, créateur de la célèbre série de jeu de rôle *Ultima*, commencera sa carrière (King & Borland, 2003). A la fin des années 80 et jusqu'au milieu des années 90, la démocratisation des réseaux *BBS*[111] a donné un autre moyen de diffusion de programmes par voie électronique. De nombreux concepteurs de jeux gagnaient alors leur vie en vendant leur réalisation sous forme de *« shareware »*. Ce modèle de distribution, apparu avec les logiciels utilitaires (Knopf & Ford, 2000), consiste à diffuser de manière gratuite une partie de son jeu vidéo. Si un joueur aime ce jeu, il peut ensuite en commander la version complète par la poste. Des sociétés comme ***Apogee / 3D Realms***, ***id Software*** ou encore ***Epic Games,*** qui font aujourd'hui partie des grands noms de l'industrie, ont débuté leur carrière grâce à ce mode de diffusion (Kushner, 2004).

1.2 ORIGINE HISTORIQUE DES « USINES A JEUX »

Historiquement, la pratique amateur du Game Design est intimement liée à l'histoire de l'informatique, en particulier à celle du développement de la micro-informatique personnelle. En effet, les spécialistes s'accordent à dire que le premier micro-ordinateur à avoir rencontré un succès commercial est l'*Altair 8800* ***(MITS, 1975)***. Sorti en 1975, cet ordinateur avait la particularité d'être vendu en kit. Il a donc rapproché de nombreux passionnés d'informatique, qui se sont rapidement regroupés au sein de « clubs » afin d'échanger des conseils

[111] Un **B**ulletin **B**oard **S**ystem est un ordinateur auquel d'autres ordinateurs peuvent se connecter par le biais d'un modem. Un fois connecté à un BBS par le réseau téléphonique, un utilisateur peut discuter avec d'autres membres, envoyer ou récupérer des fichiers, jouer à des jeux en réseau… Il s'agit en quelque sorte de l'ancêtre d'Internet et de ses nombreux services. Les BBS rencontreront un certain succès auprès des passionnés d'informatique, à une époque où le « réseau des réseaux » n'était pas encore accessible au grand public (soit de la fin des années 1970 au milieu des années 1990).

d'assemblage. Mais ces clubs étaient aussi un moyen d'échanger des programmes pour cette machine. L'un des rares programme disponibles lors de sa sortie était l'*Altair Basic* ***(Micro-Soft, 1975)***, première création commerciale d'une jeune société qui sera plus tard connue sous le nom de ***Microsoft***. Les détenteurs d'un *Altair 8800* pouvaient donc écrire leurs propres programmes en langage *BASIC*. Si seuls certains de ces clubs deviendront célèbres, à l'image du *Homebrew Computer Club* fréquenté par ***Steve Jobs*** et ***Steve Wozniak***, tous avaient la particularité de permettre la circulation de nombreux petits jeux vidéo écrit en *BASIC* (Wolf, 2007). Réalisés par des amateurs, ces jeux vidéo s'échangeaient directement sous forme de listing *BASIC*. Cette pratique informelle a rapidement été accompagnée par la publication d'ouvrages regroupant des codes sources de jeux en *BASIC*, tel le célèbre *Basic Computer Games* (Ahl, 1978). Ecrit par le journaliste et spécialiste de l'histoire du jeu vidéo ***David Ahl***, cet ouvrage rassemble les codes sources de 101 jeux. Il a connu plusieurs éditions pour diverses machines. La première date de 1973 et est consacrée au *BASIC* tournant sur les ordinateurs de la famille *PDP* du constructeur ***DEC***. La version de 1978, consacrée au *BASIC* de ***Microsoft***, rencontra un énorme succès auprès du public puisque plus d'un million d'exemplaires furent vendus au total. L'ouvrage fut même traduit en français par ***Sybex*** en 1980. A noter que l'année 2010 vit une réédition de cet ouvrage pour le langage *Smallbasic*[112]. Avec la sortie de nouveaux modèles de micro-ordinateurs, tels que l'*Apple II* ***(Apple, 1977)***, la création et l'échange de jeux amateurs sous forme de codes sources se développa considérablement. Certains magazines se spécialisèrent même dans cette activité, à l'image du français *Hebdogiciel* ***(Shift Éditions, 1983-1987)***. Au-delà de ce préambule, les exemples d'usines à jeux historiquement significatives sont exposés dans cette thèse *(p.166)*.

1.3 ORIGINE HISTORIQUE DES « EDITEUR DE NIVEAUX »

Une fois que le joueur crée un nouveau niveau, il souhaite en général le sauvegarder pour pouvoir le retravailler, ou tout simplement pour le diffuser. Nous avons déjà exposé quelques exemples d'éditeurs de niveaux, mais tous sont destinés à des jeux sur ordinateurs, et non à des jeux sur consoles *(p.50)*. Cela s'explique par le fait que la plupart des consoles portables ou de salon étaient, jusqu'aux milieu années 2000, dépourvues de moyens simples pour stocker et échanger des informations entre joueurs. Les éditeurs de niveaux se sont donc plus facilement développés sur ordinateur. Une des rares exceptions en la matière est *Excite Bike* ***(Nintendo, 1984),*** qui propose un éditeur de circuits alors qu'il s'agit d'un jeu pour la console *N.E.S.* ***(Nintendo, 1983)***. Mais ce jeu ne permet d'enregistrer son travail que si le joueur possède le *Famicom Data Recorder* ***(Nintendo, 1984)***, un périphérique optionnel qui permet de brancher un lecteur de cassettes à la console. Cet éditeur de circuit a d'ailleurs sciemment été incorporé à ce jeu pour promouvoir l'utilisation de ce matériel. Cependant, tous les logiciels permettant de créer de nouveaux niveaux ne sont pas forcément le fruit des créateurs originels du jeu. Tels que nous l'avons déjà évoqué *(p.50)*, il arrive que certains amateurs créent eux-mêmes des outils officieux permettant d'éditer des niveaux. Là encore, cette pratique est largement plus répandue sur ordinateur que sur console, étant donné qu'il est difficile d'ajouter du contenu à un jeu console. Une des rares exceptions en la matière est le jeu *New Super Mario Bros. Wii* ***(Nintendo, 2009)*** qui a bénéficié de deux éditeurs de niveaux non officiels moins d'un mois après sa sortie : *Reggie* ***(Trekkie, 2009)*** et *Tanooki* ***(Virus & Vash, 2009)***.

Dans un autre registre, les éditeurs de niveaux n'ont d'utilité que si des joueurs s'en servent effectivement pour créer du contenu additionnel pour un jeu donné. Comme pour les usines à jeux *(p.172)*, le rôle des communautés de créateurs est ici primordial. Pour les premiers éditeurs de niveaux, les joueurs n'avaient pas à leur disposition de moyens de communication très efficaces. Mais avec l'arrivée des *BBS* et surtout d'Internet ils ont pu se rassembler afin

[112] http://smallbasiccomputergames.com/

de s'échanger leurs créations. Il est d'ailleurs intéressant de noter que la plupart des communautés de créateurs sont nées de manière spontanée de la part des joueurs, bien avant que les éditeurs de jeux ne se prennent conscience du phénomène et l'incorporent au cycle de vie de leurs produits. Un des exemples les plus significatifs en la matière est la série des *Trackmania* ***(Nadeo, 2003-2010)***. Il s'agit de jeux de course proposant un éditeur de niveau ayant fait la renommée de la série. Les joueurs les plus créatifs peuvent s'échanger leurs créations à travers un service baptisé *Trackmania Exchange* ***(2005-2011)***. Mais ce service n'est pas directement intégré au jeu : les joueurs doivent en sortir pour s'échanger du contenu. Cela traduit l'origine de cette fonctionnalité, qui est née en tant que site d'échange construit par des amateurs. Le succès de ce site a poussé les développeurs du jeu à l'adouber comme un « service officiel », bien qu'il ne soit pas une partie intégrante du jeu. Néanmoins, les quelques 262.764 circuits[113] hébergés par ce site sont régulièrement consultés par les joueurs qui hébergent des serveurs pour les parties multijoueurs. En effet, si un circuit créé par un joueur est installé sur un serveur, il sera automatiquement partagé à tous les joueurs qui s'y connecteront pour jouer. Un principe similaire était déjà utilisé par le jeu *Half-Life* ***(Valve Software, 1998)*** pour diffuser les cartes créées par les joueurs. Ce type de méthode de partage a posé les bases des plateformes de partage directement intégrées au jeu, comme nous le verrons avec le **« Jeu 2.0 »** *(p.200)*.

D'un point de vue historique, que ce soit pour le partage de créations, pour la mise à disposition d'outils officieux comme officiels, ou tout simplement pour la démocratisation de la modification de jeux commerciaux par les joueurs, le titre le plus marquant est sans conteste *Doom* ***(id Software, 1993)***. En effet, ses concepteurs furent au départ très surpris lorsque de nombreux fans créèrent de nouveaux niveaux et des versions modifiées de leur titre *Wolfenstein 3D* ***(id Software, 1992)***. Ces créations s'échangeaient principalement par les serveurs *BBS*, déjà utilisés par ***id Software*** pour commercialiser son jeu selon le principe du *« shareware »* *(p.47)*. Au lieu de réprimander les joueurs qui avaient « hacké » leur titre pour le modifier et créer un éditeur de niveau officieux, ***id Software*** chercha au contraire à encourager cette pratique avec *Doom*, alors en cours de réalisation. Concrètement, le programmeur du jeu, ***John Carmack***, a inventé le format de fichier *WAD (acronyme de « Where's All the Data? »)* afin de séparer le « contenu » du jeu de son « moteur ». Cela permit aux amateurs de créer du contenu pour ce jeu sans avoir à hacker le jeu, et représente donc un des premiers exemples de légitimation de la modification des jeux commerciaux par les professionnels de l'industrie. Selon ***Laukkanen*** (2005) cette approche consistant à séparer le « moteur » de jeu de son « contenu » pose même les bases du *« middleware »* *(p.46)*. Mais cette légitimation ne s'est pas faite sans contrepartie. La seule requête d'***id Software*** fut que le contenu créé par les joueurs ne puisse fonctionner que sur la version complète payante du jeu, et non sur la version de démonstration gratuite *(principe du « shareware »)*. Le hasard fit que la sortie de ce titre, qui rencontrait déjà un succès critique et commercial considérable, s'accompagne du développement d'Internet pour le grand public. Ainsi, les amateurs qui créent de nouveaux niveaux pour *Doom* peuvent les échanger à travers des sites de fans tels que *Doomworld* ***(1993-2011)*** et *Gamers.org-DoomGate* ***(1994-2011)***. Pourtant, ***id Software*** ne livra pas ses outils de développement avec *Doom*, bien qu'elle encouragea les joueurs à modifier son titre. En effet, l'éditeur de niveaux utilisé pour créer *Doom* ne tourne que sur les ordinateurs du constructeur *NeXT*, alors que le jeu tourne sous *PC (Ms-Dos)*. Les outils de développement n'étant pas accessibles aux joueurs pour de simples questions techniques, ceux-ci s'empressèrent de créer de nombreux éditeurs de niveaux. Ainsi, *Doom Editing Utility* ***(Brendon Wyber, 1994)*** sera diffusé un mois après la sortie du jeu (Kushner, 2004). Il faudra ensuite attendre la sortie de *Duke Nukem 3D* ***(3D Realms, 1996)*** pour qu'un développeur livre directement tous ses outils de développement avec le jeu. D'après ***Scott Miller***, dirigeant de ***3D Realms***, cette décision était courageuse à une époque où les studios de l'industrie

[113] Relevé le 13-03-11 à partir de http://www.tm-exchange.com/

pensaient qu'il ne fallait absolument pas diffuser ses outils et technologies (Djaouti, 2010b, 2010c). L'histoire montrera qu'à l'inverse, cette pratique augmente le potentiel commercial du jeu en prolongeant sa durée de vie. Ainsi, plus de quinze ans après sa sortie, *Doom* est toujours joué et utilisé comme support de création. De nouveaux éditeurs de niveaux officieux continuent même à voir le jour pour ce titre, à l'image *Doom Builder 2* ***(Pascal vd Heiden, 2009)***.

1.4 ORIGINE HISTORIQUE DU « MODDING »

D'un point de vue historique, l'esprit même du « modding » semble remonter aux pionniers de l'histoire du jeu vidéo, et plus précisément à *Spacewar !* (Steve Russell & al., 1962). La première version de ce jeu inventé par des étudiants du ***MIT*** fut programmée par ***Steve Russel***. Elle fut ensuite modifiée par ***Peter Samson***, qui modifia le fond étoilé du jeu afin qu'il reflète la véritable position des étoiles. ***Dan Edwards*** ajouta alors un soleil central, ce qui transforma considérablement l'expérience de jeu. Ces changements furent complétés par la possibilité de faire des « sauts hyper-spatiaux » que rajouta ***Martin Graetz***. De telles modifications du jeu originel furent possible par sa nature « open-source » : tout le monde avait accès au code source du jeu et pouvait donc le modifier à sa guise (Graetz, 1981). Mais ces transformations eurent lieu dans un espace clos *(le Tech Model Railroad Club au **MIT**)*, les « moddeurs » pouvaient alors facilement poser des questions d'ordre technique au créateur originel du jeu.

Le premier mod effectué sans l'aide (ni le consentement) des développeurs originel d'un jeu vidéo semble pour l'instant être *Dino Smurfs* ***(Andrew Johnson & Preston Nevins, 1983)***, un « mod » de *Dino Eggs* ***(David Schroeder, 1983)***. Il fut rapidement suivi par *Castle Smurfenstein* ***(Andrew Johnson & Preston Nevins, 1983)***, basé sur le jeu *Castle Wolfenstein* ***(Silas Warner, 1981)***. Ces « mods » furent créés par deux adolescents qui remplacèrent les graphismes et les sons de leurs jeux vidéo favoris par des éléments provenant de la bande dessinée *Les Schtroumpfs* ***(Peyo, 1958)***. Ce qui débuta comme une blague potache demanda concrètement aux deux créateurs amateurs d'étudier la manière dont les graphismes et sons étaient stockés, de manière à pouvoir les remplacer. Et ce, sans aucune indication de la part des créateurs originels (Wagner, 2002). L'auteur du jeu *Dino Eggs* n'était d'ailleurs même pas au courant de l'existence d'un tel « mod » pour son jeu avant qu'il eu l'occasion de le tester en 1998 (A. Johnson, 1999). Dans un esprit similaire, un exemple antérieur nous vient du jeu d'arcade *Crazy Otto* ***(General Computer Corporation, 1981)***. Il s'agit d'une version modifiée du jeu *Pac-Man* ***(Namco, 1980)*** qui ajoute des jambes au célèbre glouton et qui modifie légèrement les paramètres de toutes les règles du jeu *(vitesse, points…)*. Les compétences techniques des employés de la société ***General Computer Corporation (GCC)*** leur permirent de créer des « cartes d'amélioration » qui se branchaient directement sur des bornes d'arcade enfin d'en modifier le jeu (Kushner, 2002). Une borne d'arcade modifiée de la sorte est appelée un *« bootleg »*, en référence aux enregistrements audio pirates de concert musicaux. En effet, la légalité d'une telle pratique fut mise à l'épreuve lorsque la société commercialisa sa première réalisation, *Super Missile Attack* ***(1981)***. ***Atari***, société créatrice du jeu originel *Missile Command* ***(1980)*** qui se voyait ainsi modifié, poursuivi ***GCC*** en justice. ***GCC*** remporta ce procès et continua donc à commercialiser librement des kits de modifications pour jeux d'arcade. Quand ***GCC*** montra *Crazy Otto* à ***Midway***, le distributeur américain de *Pac-Man* qui attendait impatiemment que le développeur ***Namco*** en sorte une suite, ils furent surpris de la réaction de l'éditeur. En effet, au lieu de les menacer de poursuites judiciaires, ***Midway*** engagea ***GCC*** afin qu'ils continuent leur travail sous la direction de l'éditeur. Cette collaboration aboutira au jeu *Ms. Pac-Man* ***(Midway, 1981)***, dont le succès commercial n'est plus à démontrer (Kent, 2001).

Après ces exemples pionniers, la pratique du modding fut clairement popularisée par les jeux de tir en vue subjective sur *PC*. Tout d'abord grâce à *Doom* et ses outils officieux tel que *DeHacked* ***(Greg Lewis, 1994)***, qui permet de modifier les règles du jeu. Ce jeu vit apparaître de nombreux « mods » et montra à l'industrie vidéoludique l'intérêt de créer des outils permettant aux joueurs de modifier leurs jeux. Quelques années plus tard, *Counter-Strike* semble être le premier « mod » dont le succès populaire dépasse largement celui de son jeu d'origine. Ce succès permet au grand public de découvrir la pratique du modding, qui restait jusqu'alors cantonnée à une certaines frange de joueurs passionnés (Laukkanen, 2005).

1.5 ORIGINE HISTORIQUE DES « MENUS D'OPTIONS »

D'un point de vue historique, les premiers exemples de ce genre d'outils semblent être les *« DIP switches »*, des interrupteurs physiques se trouvant sur les bornes d'arcade du début des années 80. Ces interrupteurs étaient utilisés par les gérants de salles d'arcade pour régler la difficulté, et donc la rentabilité, des jeux qu'ils exploitaient. Avec l'apparition de mémoires *RAM* alimentées peu coûteuses dans les années 90, ces interrupteurs furent remplacés par des menus graphiques, dont l'accès était toujours strictement réservé aux gérants de salles d'arcade. Lorsque les jeux vidéo migrèrent des salles d'arcade aux consoles de salon, ces menus furent rendus accessibles aux joueurs, et évoluèrent pour correspondre à leurs attentes. Au lieu de régler seulement la difficulté du jeu, il fut donc rapidement possible de configurer les modes de contrôles, l'affichage, le son… A noter également que la plupart des jeux tournant sur la console *VCS 2600* ***(Atari, 1977)*** proposaient plusieurs « modes de jeu ». Les joueurs pouvaient passer d'un mode de jeu à l'autre en appuyant sur la touche *« Game Select »* de la console. Nous pouvons voir là une manière primitive de permettre aux joueurs de « configurer » leurs jeux, et donc une source d'influence pour les « menu d'options » actuels.

1.6 ORIGINE HISTORIQUE DU « JEU 2.0 »

Si le « Jeu 2.0 » possède bien quelques différences avec les « usines à jeux », force est de constater qu'ils ont beaucoup de points en commun *(p.201)*. Cette ressemblance se retrouve également dans l'histoire respective de ces deux catégories d'outils.

Au sein de notre corpus de « Jeu 2.0 », 31 éléments ont une date de publication comprise entre 2007 et 2011. Nous pouvons donc ici confirmer le caractère relativement récent du « Jeu 2.0 » qui avait été exposé lors de l'étude de sa définition *(p.200)*. Mais notre corpus comporte également 4 exemples d'applications plus anciennes, la doyenne datant de 1994. Il convient de préciser qu'à cette époque la notion de « Jeu 2.0 » n'était pas évoquée. Et ce pour une raison très simple : le terme même de « Web 2.0 », qui a inspiré l'expression « Jeu 2.0 », date de 2005 (O'Reilly, 2005). Cependant, même si la notion de « Jeu 2.0 » n'existait pas encore, pas plus que l'engouement pour le *« player-generated-content »*, nous recensons dans notre corpus quatre applications qui correspondent à la définition actuelle du « Jeu 2.0 ». Nous nous retrouvons ici dans une situation ressemblant à celle décrite pour le secteur du « Serious Game ». Même si l'emploi massif du terme « Serious Game » est récent, il existe de nombreux exemples de jeux plus anciens qui correspondent à sa définition actuelle. Nous les avions alors qualifiés « d'ancêtres » des Serious Games *(p.33)*. Il nous semble donc tout naturel de faire de même ici, et de parler « d'ancêtres » de « Jeu 2.0 ».

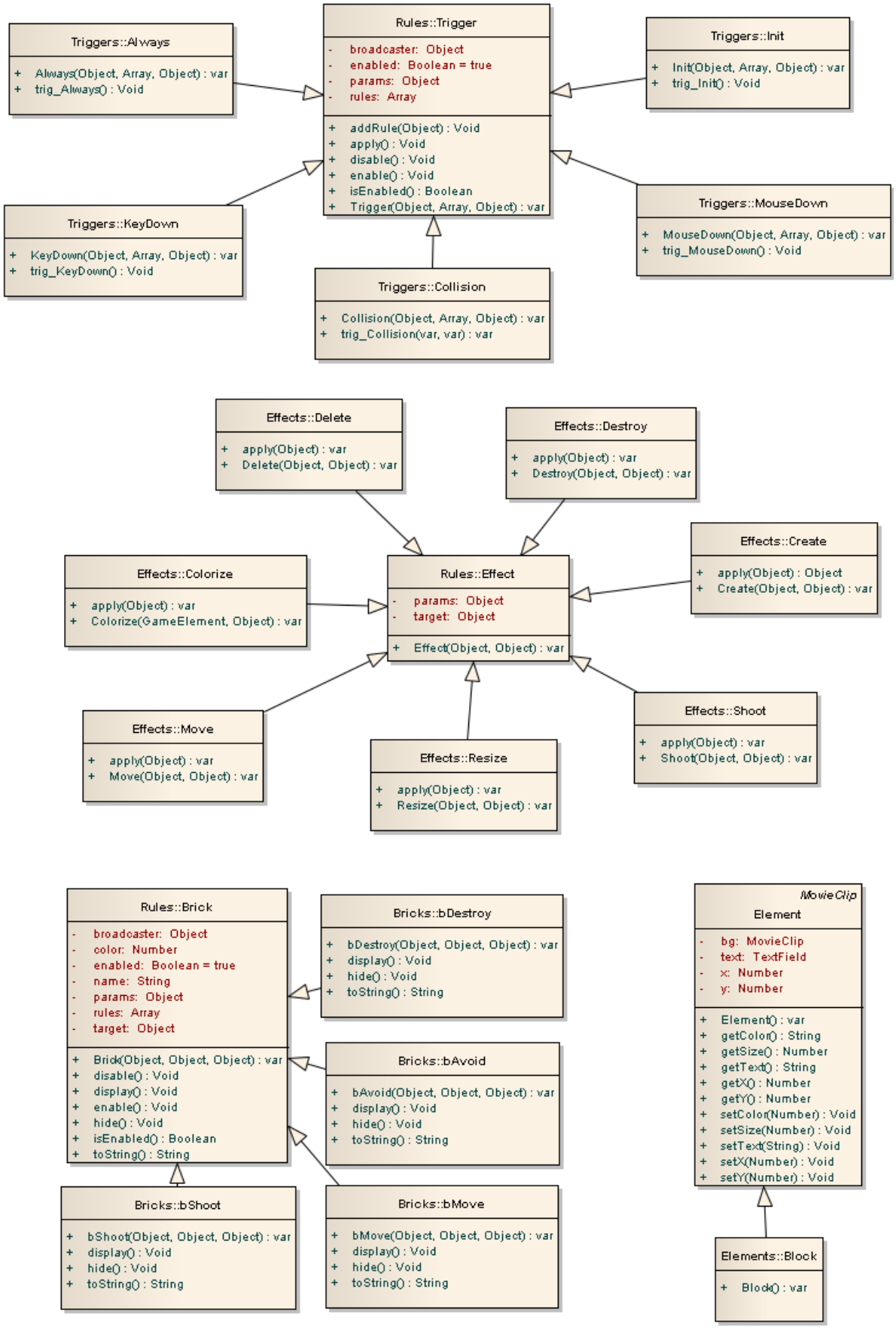
Si nous étudions ces ancêtres de « Jeu 2.0 » en détail, nous remarquons que le plus ancien exemple de notre corpus, *Addventure*, date de 1994. Pour être plus précis, les premières versions d'*Addventure* furent écrites entre 1987 et 1988 par ***Allen S. Firstenberg*** alors qu'il était étudiant à la *Nyack High School*. Il s'agissait d'un jeu d'aventure en mode texte dans

lequel les joueurs pouvaient rajouter des éléments au fur et à mesure qu'ils jouaient. Ce jeu et ses outils de création bénéficient du serveur *BBS* de ce lycée pour proposer une plateforme de partage. L'arrêt du serveur *BBS* signera la fin du jeu et de ses outils. L'arrivée d'Internet pour le grand public lui donnera une seconde vie à travers une version Internet diffusée à l'échelle mondiale dès 1994, intitulée *Addventure 1.0* (Firstenberg, 1997). Suivront ensuite *Addventure 2.0* en 1995 et *Addventure 2.1* en 1997, toujours développées par ***Allen S. Firstenberg***. Le jeu étant hébergé par l'auteur du logiciel lui-même, il ne propose pas la possibilité de créer de nouvelles histoires, mais « seulement » de compléter une histoire interactive initiée par ***Allen S. Firstenberg***. Concrètement, le jeu est divisé en différents « épisodes », qui offrent chacun une description textuelle et plusieurs choix de navigation. Pour chaque « épisode » existant, le joueur peut décider de créer une nouvelle connexion vers un nouvel « épisode » qu'il rédigera lui-même. L'histoire se construit ainsi progressivement pour chaque joueur qui décide de l'enrichir. La première histoire d'*Addventure*, écrite selon se principe entre 1994 et 1995, totalisera plus de 10.000 « épisodes » créées par les joueurs. Deux nouvelles histoires furent alors initiées par l'auteur du logiciel *Addventure*. Elles totaliseront respectivement 41.119 « épisodes »[114] et 35.253 « épisodes »[115], avant d'être fermées aux contributions en 1998 à cause d'un manque d'espace d'hébergement. Cette initiative pionnière est loin d'être la seule en son genre, car elle sera rapidement suivie par d'autres outils de création de jeux d'aventure textuels. Par exemple, *StorySprawl* arrivera sur Internet en 1998 et proposera cette fois à tout internaute de créer de nouvelles histoires par lui-même. Au-delà de cette innovation, la logique de création est très proche de celle proposée par *Addventure*. En plus de *StorySprawl*, le site Internet d'*Addventure* recense près d'une vingtaine d'applications similaires[116]. Si toutes proposent un système de création collaborative, les modalités varient de l'application logicielle directement contrôlée par l'utilisateur, à l'image d'*Addventure*, à des systèmes où un auteur central évalue des contributions reçues par e-mail avant d'éventuellement les intégrer à l'histoire. Bien que méconnue, la flamme pionnière de création de jeux d'aventure et autres fictions en mode texte de manière collaborative perdure de nos jours. Ainsi, en 2005 sortit *StoryLand*, qui fut un digne successeur spirituel d'*Addventure* et *StorySprawl* bien qu'il reposaient sur des technologies plus modernes. La création de tels systèmes collaboratifs en ligne permettant de créer et partager des jeux d'aventures en mode texte est même un sujet de recherche scientifique en informatique. Par exemple, ***Chou*** expose à travers un mémoire la réalisation de la plateforme *B.A.S.S.* (Chou, 2005). En exposant l'état de l'art préalable à la création de son logiciel, cet universitaire analyse plusieurs usines à jeux et pointe du doigt le fait qu'elles sont dépourvues de plateforme de partage intégrées, ce qui à ses yeux limite le potentiel d'échange que peut apporter l'acte créatif. Pour en revenir à notre corpus, son dernier exemple d'ancêtre de « Jeu 2.0 », *DUNG*, est un outil spécialisé dans la création de jeux de rôle et d'aventure avec des graphismes 2D. Il fut créé par un duo d'amateurs passionnés, ***Dan Bradley*** et ***Tom Hehre***, alors qu'ils étaient étudiants à l'université. Leur initiative rencontra suffisamment de succès auprès des joueurs pour leur permettre de fonder une petite société afin de faire évoluer leur outil (Hehre, 2000). Ce dernier connaît toujours un certain succès aujourd'hui sous le nom de *BYOND*.

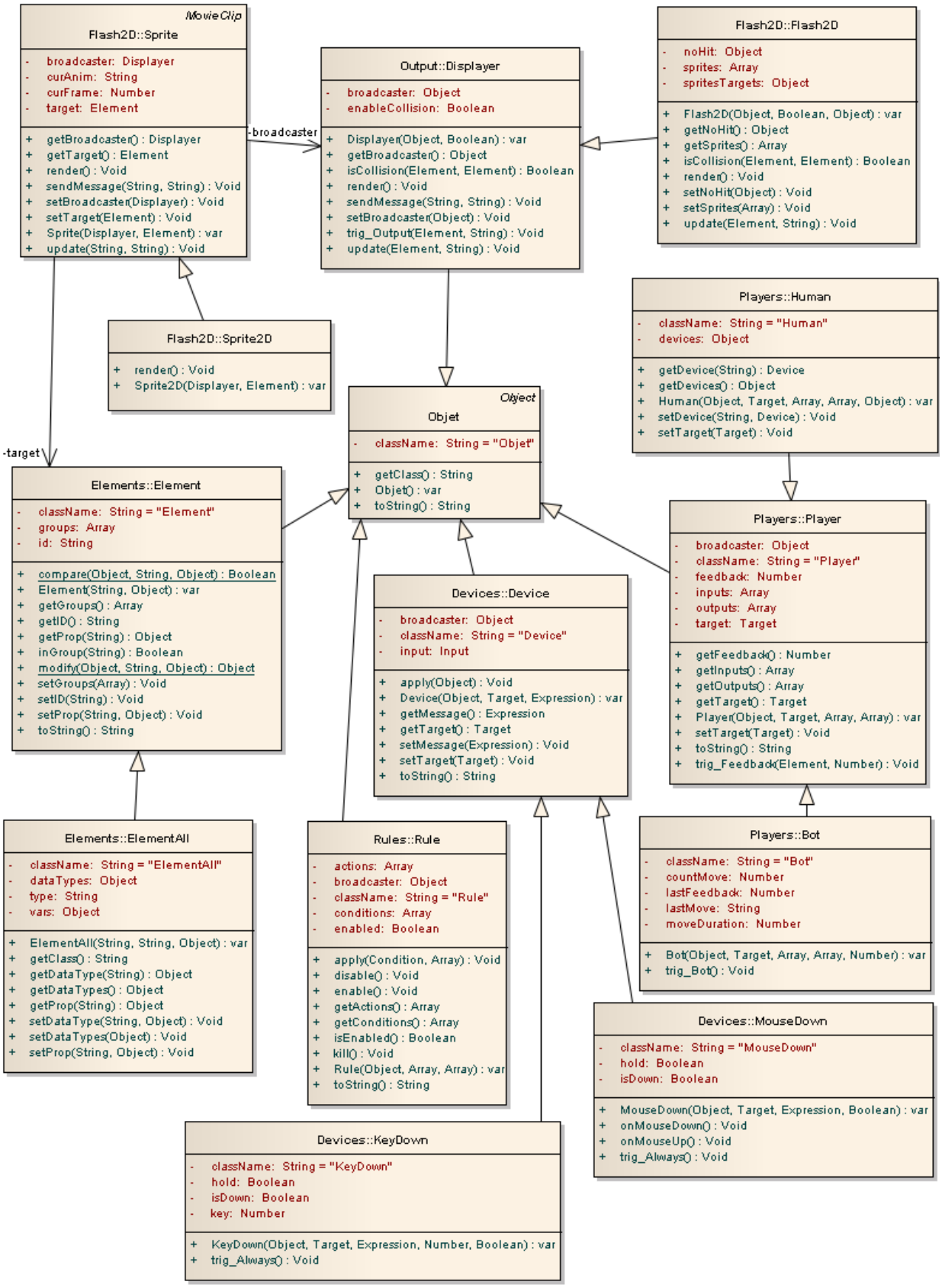
En résumé, il était donc possible sur Internet de créer et partager des jeux d'aventure s'appuyant sur un mode de représentation textuel dès 1994, puis graphique à partir de 1996. Il suffisait pour cela d'utiliser ces quelques outils pionniers qui préfiguraient le courant du « Jeu 2.0 ». Nous pouvons remarquer ici une certaine similarité entre les débuts de l'histoire du « Jeu 2.0 » et celui des « usines à jeux » *(p.49)*. En effet, les usines à jeux ont également débutées avec l'apparition d'outils permettant de créer des jeux d'aventure en mode texte, suivis quelques années plus tard par des outils permettant de créer d'autres genres de jeux dont des aventures graphiques. Si l'histoire des usines à jeux débute par l'apparition d'outils

[114] Relevé le 07-03-11 sur http://www.addventure.com/cgi-bin/about2/addventure/game2/episode/?gamestat
[115] Relevé le 07-03-11 sur http://www.addventure.com/cgi-bin/about2/addventure/game3/episode/?gamestat
[116] Relevé le 07-03-11 à partir de http://www.addventure.com/addventure/others.html

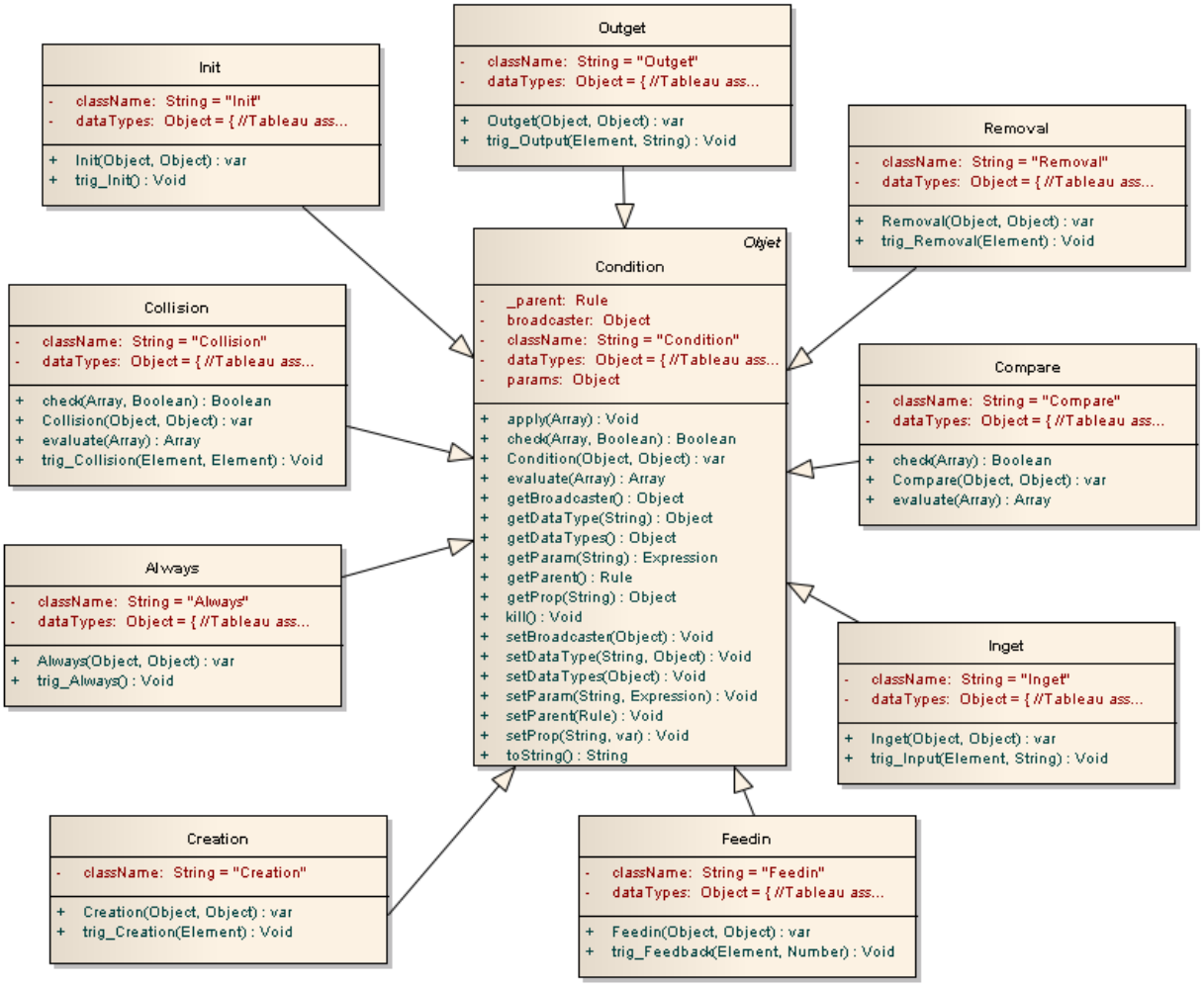
spécialisés dans un certain genre de jeux avant l'apparition d'outils généralistes, il en va de même pour le « Jeu 2.0 ». Quant au pourquoi du genre du jeu d'aventure en mode texte comme berceau des premiers outils techniques de création vidéoludique, nous pouvons penser que c'est lié à deux facteurs. Tout d'abord, une représentation à base de texte reste plus facile à gérer que des images en mouvement d'un point de vue technique. Les premiers outils de créations vidéoludique étant le fruit d'amateurs passionnés, il semble logique que leurs ressources limitées les aient conduits vers un genre de jeu abordable techniquement. Mais aussi, la création de jeux d'aventure en mode texte se résume généralement à l'écriture de textes liés entre eux. Si l'on excepte la phase de conception, la réalisation d'un tel jeu est donc relativement rapide, et donc plus facilement à la portée de créateurs amateurs. Toutes les conditions semblaient donc réunies pour faire du jeu d'aventure en mode texte le berceau des outils techniques de création vidéoludique, en tout cas pour les outils permettant de créer de nouveaux jeux autonomes.



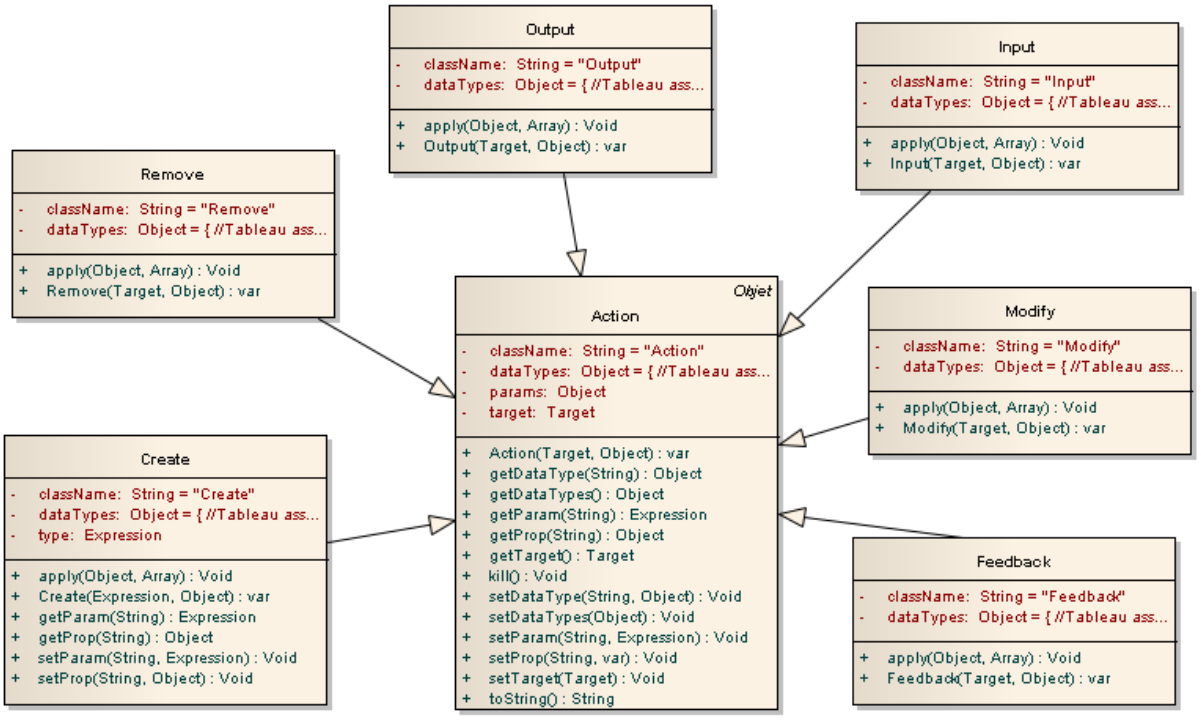
105. *Gam.B.A.S. 1.0* : schéma de toutes les classes du prototype #1



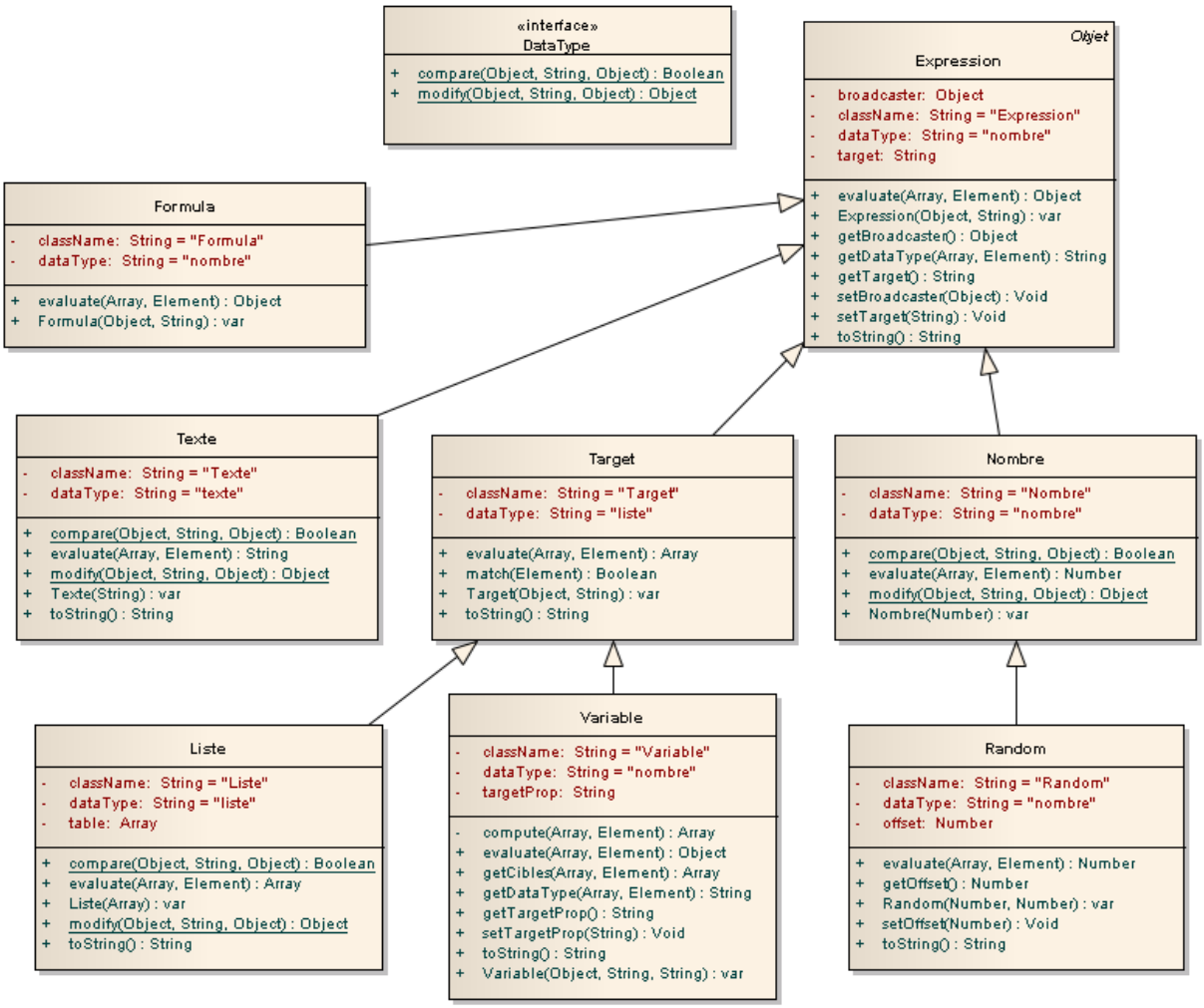
106. *Gam.B.A.S. 2.0* : schéma des classes « *Elément* », « *Règle* », « *Afficheur* » et « *Joueur* »



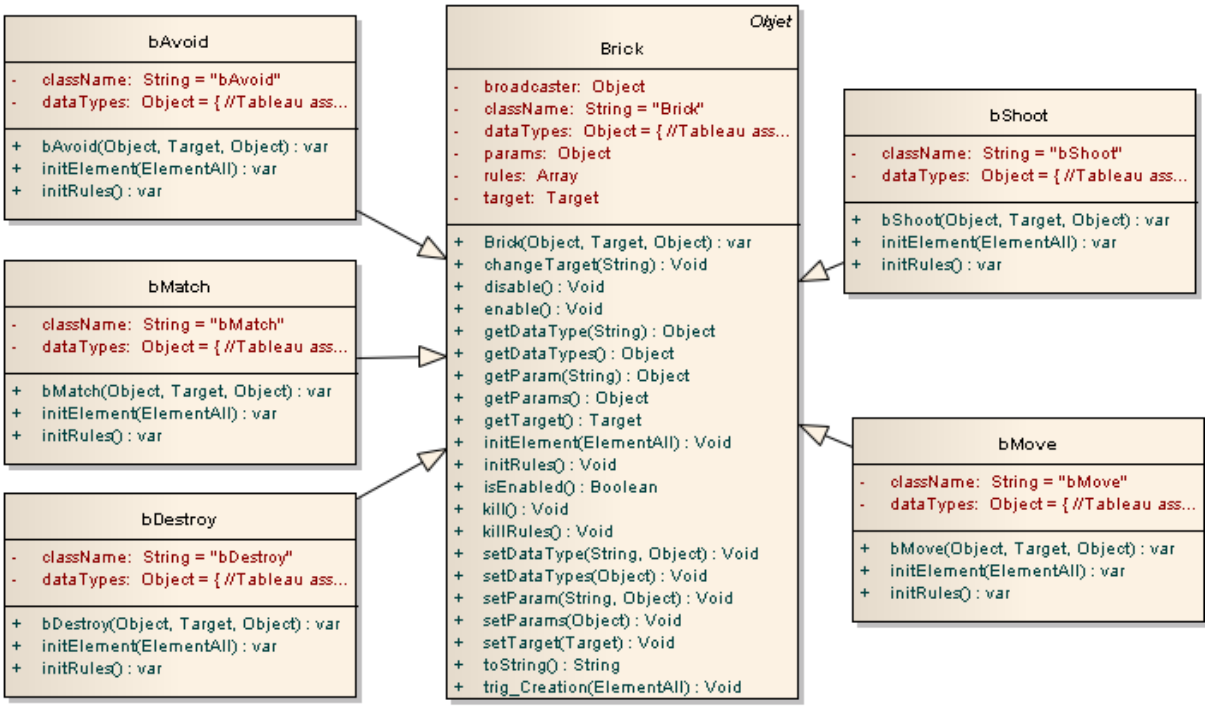
107. *Gam.B.A.S. 2.0* : schéma des classes *« Condition »*



108. *Gam.B.A.S. 2.0* : schéma des classes *« Action »*

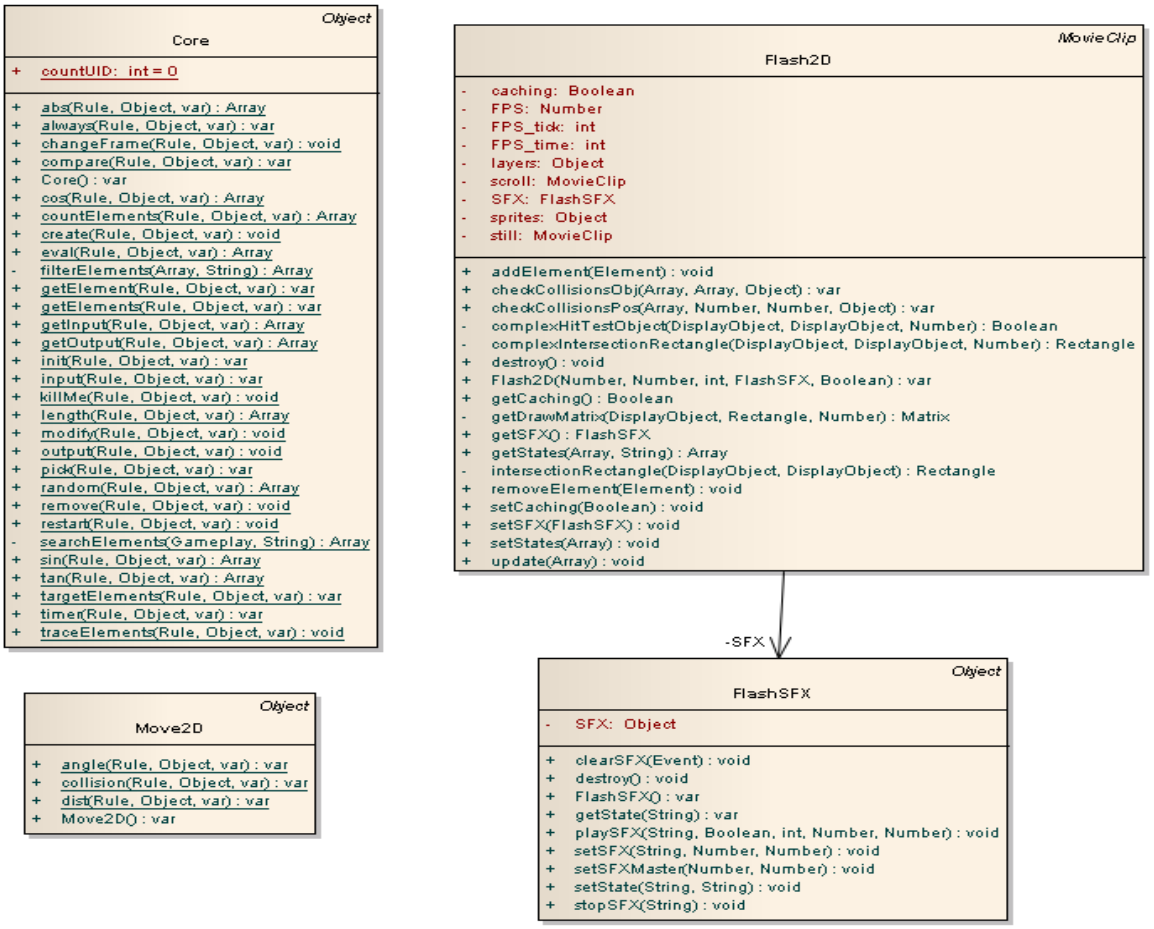


109. *Gam.B.A.S. 2.0* : schéma des classes *« Expression »*



110. *Gam.B.A.S. 2.0* : schéma des classes *« Brique »*

111. *LudoForge* : schéma des classes « *Gameplay* », « *Règle* », « *Elément* » et « *Joueur* »



112. *LudoForge* : schéma des classes « *Noyau* » et « *Afficheur* »

BIBLIOGRAPHIE

1 REFERENCES

Aarseth, E., Smedstad, S. M., & Sunnana, L. (2003). A multidimensional typology of games. In C. Marinka & R. Joost (Eds.), *Level Up Conference Proceedings: Proceedings of the 2003 Digital Games Research Association Conference*. Utrecht: University of Utrecht. Retrieved from http://www.digra.org/dl/display_html?chid=05163.52481.pdf

Abt Associates. (2005). Biography of Clark C. Abt. Retrieved October 26, 2010, from http://www.abtassociates.com/page.cfm?PageID=104

Abt, C. C. (1970). *Serious Games*. Viking Press.

Abt, C. C., Hodder, J. C., & O'Sullivan, T. C. J. (1965). *TEMPER: The Theory of the Model*. Raytheon.

Adams, E. (2001, February 2). Dogma 2001: A Challenge to Game Designers. Retrieved June 14, 2010, from http://www.designersnotebook.com/Columns/037_Dogma_2001/body_037_dogma_2001.htm

Adams, E. (2009). *Fundamentals of Game Design* (2 ed.). New Riders Press.

Ahl, D. H. (1978). *BASIC Computer Games: Microcomputer Edition* (Microcomputer ed.). Workman Pub Co.

Albinet, M. (2010). *Concevoir un jeu vidéo. Tout ce que vous devez savoir pour élaborer un jeu vidéo* (1er ed.). FYP éditions.

Alvarez, J. (2007, December 17). *Du jeu vidéo au serious game, approches culturelle, pragmatique et formelle* (PhD Thesis). Toulouse, France: Université de Toulouse.

Alvarez, J., Alvarez, V., Djaouti, D., & Michaud, L. (2010). *Serious Games: Training & Teaching - Healthcare - Defence & security - Information & Communication*. IDATE.

Alvarez, J., & Djaouti, D. (2010). *Introduction au serious game*. Questions Théoriques.

Alvarez, J., & Maffiolo, V. (2011). Etude de l'impact de communications électroniques basées sur le Serious game. *Revue de l'Electricité et de l'Electronique*, *2011*(4).

Alvarez, J., & Michaud, L. (2008). *Serious Games : Advergaming, edugaming, training...* IDATE.

Alvarez, J., Rampnoux, O., Jessel, J., & Methel, G. (2007). Serious Game: just a question of posture? In *Proceedings of Artificial & Ambient Intelligence Convention (AISB'07)*. Presented at the Artificial & Ambient Intelligence Convention (AISB'07), Newcastle upon Tyne, United Kingdown.

Amato, E. A. (2008, November 25). *Le jeu vidéo comme dispositif d'instanciation : du phénomène ludique aux avatars en réseau*. Paris, France: Université Paris 8.

Amory, A. (2007). Game Object Model Version II: A Theoretical Framework for Educational Game Development. *Educational Technology Research and Development*, *55*(1), 51-77.

Andersen, E., Liu, Y., Apter, E., Boucher-Genesse, F., & Popović, Z. (2010). Gameplay analysis through state projection. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games* (pp. 1-8). Monterey, California: ACM. doi:10.1145/1822348.1822349

Anderson, J. (1983). Who Really Invented The Video Game? *Creative Computing Video & Arcade Games*, *1*(1), 8.

Annetta, L., & Cheng, M. (2008). Why Educational Video Games? In L. A. Annetta (Ed.), *Serious Educational Games: From Theory to Practice* (pp. 1-12). Sense Publishers.

Apperley, T. H. (2006). Genre and game studies: Toward a critical approach to video game genres. *Simulation & Gaming*, *37*(1), 6 -23. doi:10.1177/1046878105282278

Argent, L., Depper, B., Fajardo, R., Gjertson, S., Leutenegger, S. T., Lopez, M. A., & Rutenbeck, J. (2006). Building a Game Development Program. *Computer*, *39*(6), 52-60.

Avedon, E. M., & Sutton-Smith, B. (1971). *Study of Games*. John Wiley & Sons.

Baer, R. H. (2005). *Videogames: In the Beginning*. Rolenta Press.

Balian, V. (2010, December 15). Gameplay : le projet. Retrieved May 2, 2011, from http://gameplay.lecube.com/le-projet/

Bartle, R. (2005). Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs (1996). In K. Salen & E. Zimmerman (Eds.), *The Game Design Reader: A Rules of Play Anthology*. The MIT Press.

Barton, M., & Loguidice, B. (2009, February 6). The History of the Pinball Construction Set: Launching Millions of Creative Possibilities. Retrieved October 27, 2010, from http://www.gamasutra.com/view/feature/3923/the_history_of_the_pinball_.php?print=1

Bateman, C., & Boon, R. (2005). *21st Century Game Design* (1er ed.). Charles River Media.

Bates, B. (2004). *Game Design* (2 ed.). Course Technology PTR.

Becker, K. (2007). Battle of the Titans: Mario vs. MathBlaster (Vol. 2007, pp. 2707-2716). Presented at the World Conference on Educational Multimedia, Hypermedia and Telecommunications 2007, Vancouver, Canada. Retrieved from http://editlib.org/p/25753

Becker, K. (2011). The Magic Bullet. *International Journal of Game-Based Learning, 1*(1), 19-31. doi:10.4018/ijgbl.2011010102

Becker, K., & Parker, J. R. (2007). Serious games + computer science = serious CS. *J. Comput. Small Coll.*, *23*(2), 40-46.

Bergeron, B. (2006). *Developing Serious Games* (1er ed.). Charles River Media.

Bjork, S., & Holopainen, J. (2004). *Patterns in Game Design* (1er ed.). Charles River Media.

Blanchet, A. (2010). *Des Pixels à Hollywood*. Editions Pix'N Love.

Bogacs, H. (2008). *Game Mods - a survey of modifications, appropriation and videogame art* (Bachelor Thesis). Austria: Vienna University of Technology.

Boudier, V. L., & Dambach, Y. (2010). *Serious Game : Révolution pédagogique*. Hermes Science Publications.

Brathwaite, B., & Schreiber, I. (2008). *Challenges for Game Designers* (1er ed.). Charles River Media.

Briggs-Myers, I., & Myers, P. B. (1980). *Gifts Differing: Understanding Personality Type*. Davies-Black Publishing.

Brougère, G. (2005). *Jouer/Apprendre*. Economica.

Brown, S. J., Lieberman, D. A., Gemeny, B. A., Fan, Y. C., Wilson, D. M., & Pasta, D. J. (1997). Educational video game for juvenile diabetes: results of a controlled trial. *Informatics for Health and Social Care*, *22*(1), 77-89.

Bruckman, A. (1999). Can Educational Be Fun? Presented at the Game Developer's Conference, San Jose, USA.

Bura, S. (2006, March). A Game Grammar. *StephaneBura.com*. Retrieved June 13, 2010, from http://www.stephanebura.com/diagrams/

Burgos, D., Moreno-Ger, P., Sierra, J. L., Fernandez-Manjon, B., Specht, M., & Koper, R. (2008). Building adaptive game-based learning resources: The integration of IMS Learning Design and. *Simulation Gaming*, *39*(3), 414-431. doi:10.1177/1046878108319595

Caillois, R. (1967). *Les jeux et les hommes* (Ed. rev. et augm.). Gallimard.

Caillois, R. (2005). Definition of Play: the Classification of Games (1962). In K. Salen & E. Zimmerman (Eds.), *The Game Design Reader: A Rules of Play Anthology*. The MIT Press.

Callahan, M. E. (2000). The History of Shareware. *Paul's Picks*. Retrieved February 25, 2011, from http://paulspicks.com/help/history_of_shareware.aspx

Chen, J., & Ringel, M. (2001). *Can Advergaming be the Future of Interactive Advertising?* <kpe>.

Chen, J. (2006). *Flow in Games* (MFA Thesis). USA: School of Cinematic Arts, University of Southren California.

Cheruette, M. (2009, June 1). *Vers une méthodologie pour la conception des Serious Games* (Master thesis). Paris, France: Université Paris XIII.

Chou, W. (2005, October 26). *Browser-based Adventure Game System for Storytelling* (Master thesis). Queensland, Australia: University of Queensland.

Church, D. (2005). Formal Abstract Design Tools (1999). In K. Salen & E. Zimmerman (Eds.), *The Game Design Reader: A Rules of Play Anthology*. The MIT Press.

Clark, K., Brandt, J., Hopkins, R., & Wilhelm, J. (2009). Making Games after School: Participatory Game Design in Non-Formal Learning Environments. *Educational Technology*, *49*(6), 40-44.

Clark, K., & Sheridan, K. (2010). Game Design Through Mentoring and Collaboration. *Journal of Educational Multimedia and Hypermedia*, *19*(2), 125–145.

Claypool, K., & Claypool, M. (2005). Teaching software engineering through game design. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education - ITiCSE '05* (p. 123). Presented at the the 10th annual SIGCSE conference, Capacrica, Portugal. doi:10.1145/1067445.1067482

Cohen, D. (2009, October 30). OXO aka Noughts and Crosses - The First Video Game. *About.com : Classic Video Games*. Retrieved October 27, 2010, from http://classicgames.about.com/od/computergames/p/OXOProfile.htm

Connor, T., Fiske, A., & Kennedy, R. (2006, April 30). *The Impact of Frame Rate and Resolution on Player Movement in First-Person Shooters* (Master thesis). USA: Worcester Polytechnic Institute.

Converse, T. (1985). GameMaker Manual. Activision.

Cook, D. (2007, July 19). The Chemistry Of Game Design. *Gamasutra*. Retrieved June 13, 2010, from http://www.gamasutra.com/view/feature/1524/the_chemistry_of_game_design.php?print=1

Cook, D. (2009, July 15). Lost Garden: Flash Love Letter. *Lost Garden*. Retrieved May 20, 2011, from http://www.lostgarden.com/2009/07/flash-love-letter-2009-part-1.html

Corti, K. (2007, October 16). Serious Games - Are We Really A Community? *Serious Games Source*. Retrieved October 29, 2010, from http://www.seriousgamessource.com/item.php?story=15832

Costikyan, G. (2005). I have no words and I must design (1994). In K. Salen & E. Zimmerman (Eds.), *The Game Design Reader: A Rules of Play Anthology*. The MIT Press.

Crawford, C. (1982). *The Art Of Computer Game Design: Reflections Of A Master Game Designer*. Osborne/McGraw-Hill,U.S.

Crawford, C. (2002). *The Art of Interactive Design: A Euphonious and Illuminating Guide to Building Successful Software* (1er ed.). No Starch Press.

Crawford, C. (2003). *Chris Crawford on Game Design*. New Riders Games.

Crawford, C. (2004). *Chris Crawford on Interactive Storytelling* (illustrated edition.). New Riders Games.

Csíkszentmihályi, M. (1990). *Flow: the Psychology of Optimal Experience*. Harper & Row.

De Buttet, S. (2010, November 5). Serious Games: des marchés multiples, avec des modèles économiques divers. *LeMonde.fr*. Retrieved November 20, 2010, from http://www.planete-plus-intelligente.lemonde.fr/villes/serious-games-des-marches-multiples-avec-des-modeles-economiques-divers-_a-13-437.html

De Prato, G., Feijoo, C., Nepelski, D., Bogdanowicz, M., & Simon, J. (2010). *Born Digital / Grown Digital: Assessing the Future Competitiveness of the EU Video Games Software Industry*. European Commission Joint Research Centre.

Deguerry, N. (2009, April 1). Les serious games, nouvelle génération de processus d'apprentissage. *Inffo Flash*, (743), 15-21.

Despont, A. (2008, February 15). Serious Games et intention sérieuse : typologie. *Symetrix eLearning*. Retrieved October 29, 2010, from http://www.elearning-symetrix.fr/blog/index.php?post/2008/02/15/Serious-Games-et-intention-serieuse-%3A-typologie

Develop. (2007, July 6). The Rise of Middleware 2.0. *Develop*. Retrieved December 10, 2010, from http://www.develop-online.net/features/13/Rise-of-Middleware-20

Djaouti, D. (2007, June). *Narration et Interaction: Relations en milieu vidéoludique* (Master thesis). Toulouse, France: Université de Toulouse.

Djaouti, D. (2010a, March 4). Les pionniers du jeu vidéo. *Pix'N Love*, (11).

Djaouti, D. (2010b, September 3). Duke Nukem 3D - part 1. *Pix'N Love*, (14).

Djaouti, D. (2010c, December 4). Duke Nukem 3D - part 2. *Pix'N Love*, (15).

Djaouti, D., Alvarez, J., Jessel, J., & Methel, G. (2008). Play, Game, World : Anatomy of a Videogame. *International Journal of Intelligent Games & Simulation (IJIGS)*, *5*(1), 35-39.

Djaouti, D., Alvarez, J., Jessel, J., Methel, G., & Molinier, P. (2008). A Gameplay Definition through Videogame Classification. *International Journal of Computer Games Technology*, *2008*(1).

Donovan, T. (2010). *Replay: The History of Video Games*. Yellow Ant Media Ltd.

Dormans, J. (2008). Visualizing Game Mechanics and Emergent Gameplay. In *Meaningful Play 2008*. Presented at the Meaningful Play 2008, Michigan State University, USA. Retrieved from http://meaningfulplay.msu.edu/proceedings2008/

Dormans, J. (2009). Machinations: Elemental Feedback Structures for Game Design. In *GAME-ON-NA 2009: 5th International North American Conference on Intelligent Games and Simulation*. Presented at the GAME-ON-NA 2009: 5th International North American Conference on Intelligent Games and Simulation, Atlanta, USA. Retrieved from http://www.jorisdormans.nl/article.php?ref=machinations

Dormans, J. (2011). *Machinations: Game Feedback Diagrams*. Retrieved from http://www.jorisdormans.nl/machinations/

Dunoyer, C. (2001, May 20). Ecrire un récit interactif. *Académie de Créteil*. Retrieved November 22, 2010, from http://www.ac-creteil.fr/lettres/tice/recit_interactif/recit_interactif.htm

Egenfeldt-Nielsen, S. (2006). Overview of research on the educational use of video games. *Digital Kompetanse*, *1*(3), 184-213.

El-Nasr, M. S., & Smith, B. K. (2006). Learning through game modding. *Comput. Entertain.*, *4*(1), 7. doi:10.1145/1111293.1111301

Elverdam, C., & Aarseth, E. (2007). Game Classification and Game Design: Construction Through Critical Analysis. *Games and Culture*, *2*(1), 3-22. doi:10.1177/1555412006286892

Esposito, N. (2005). A Short and Simple Definition of What a Videogame Is. In D. C. Suzanne & J. Jennifer (Eds.), *Changing Views: Worlds in Play: Proceedings of the 2005 Digital Games Research Association Conference* (p. 6). Vancouver: University of Vancouver. Retrieved from http://www.digra.org:8080/Plone/dl/display_html?chid=06278.37547.pdf

Falstein, N. (2006, March 18). The 400 Project - Rule List. Retrieved June 14, 2010, from http://www.theinspiracy.com/Current%20Rules%20Master%20List.htm

Firstenberg, A. S. (1997, February 4). History of Addventure. Retrieved March 7, 2011, from http://www.addventure.com/cgi-bin/history.phtml/addventure/game2/episode

Frasca, G. (2003). Simulation versus Narrative: Introduction to Ludology. In M. J. P. Wolf & B. Perron (Eds.), *The Video Game Theory Reader* (1er ed.). Routledge.

Freud, S. (1985). Le créateur littéraire et la fantaisie. In *L'inquiétante étrangeté et autres essais* (pp. 29-45). Gallimard.

Fullerton, T. (2008). *Game Design Workshop, Second Edition: A Playcentric Approach to Creating Innovative Games* (2 ed.). Morgan Kaufmann.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software* (1er ed.). Addison-Wesley Professional.

Garcia-Mateos, G., & Fernandez-Aleman, J. (2009). Make Learning Fun with Programming Contests. In Z. Pan, A. Cheok, W. Müller, & A. Rhalibi (Eds.), *Transactions on Edutainment II*, Lecture Notes in Computer Science (Vol. 5660, pp. 246-257). Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-03270-7_17

Gee, J. P. (2007). *What Video Games Have to Teach Us About Learning and Literacy. Second Edition: Revised and Updated Edition* (2 ed.). Palgrave Macmillan.

Genvo, S. (2006, October 27). *Le game design de jeux vidéo : approche communicationnelle et interculturelle*. Metz, France: Université de Metz.

Genvo, S. (2008). Caractériser l'expérience du jeu à son ère numérique : pour une étude du "play design". Presented at the Colloque "Le jeu vidéo : une expérience multidimensionnelle", congrès de l'Association Francophone pour le Savoir (ACFAS), Québec, Canada.

Gillet, S., & Gorges, F. (2008). La naissance de Lode Runner. In *Pix'N Love #4*. Editions Pix'N Love.

Graetz, M. (1981, June). The Origin of Spacewar! *Creative Computing*, 56-67.

Graham, B. (1996). *Serious games: Art, interaction, technology*. Barbican Art Gallery in association with Tyne & Wear Museums.

Gray, O. (2008). Virtuoso: User Guide.

Gray, O., & Young, M. (2007). Video Games: A New Interface for Non-Professional Game Developers. Presented at the Computer/Human Interaction (CHI) 2007, San Jose, USA.

Grünvogel, S. M. (2005). Formal Models and Game Design. *Game Studies*, *5*(1).

Gudmundsen, J. (2006, May 19). Movement aims to get serious about games. *USA Today*. USA. Retrieved from http://www.usatoday.com/tech/gaming/2006-05-19-serious-games_x.htm

Habgood, J. (2007, July). *The effective integration of digital games and learning content* (PhD Thesis). England: University of Nottingham.

Habgood, J., Ainsworth, S., & Benford, S. (2005). Intrinsic Fantasy: Motivation and Affect in Educational Games Made by Children. Presented at the 12th International Conference on Artificial Intelligence in Education (AIED 2005), Amsterdam, Netherlands.

Hagen, U. (2010). Designing for Player Experience: How Professional Game Developers Communicate Design Visions. In L. Petri, T. A. Mette, V. Harko, & W. Annika (Eds.), *Proceedings of DiGRA Nordic 2010: Experiencing Games: Games, Play, and Players*. Stockholm: University of Stockholm. Retrieved from http://www.digra.org:8080/Plone/dl/display_html?chid=10343.03567.pdf

Hague, J. (1997). *Halcyon Days*. James Hague. Retrieved from http://www.dadgum.com/halcyon/

Harfield, M. (2008). *Not Dark Yet: A Very Funny Book About a Very Serious Game*. Loose Chippings Books.

Harmonie Magazine. (2009, November). Des jeux vidéo pour guérir. *Harmonie Magazine*, (267), 17.

Hayes, E. R., & Games, I. A. (2008). Making Computer Games and Design Thinking. *Games and Culture*, *3*(3-4), 309 -332. doi:10.1177/1555412008317312

Hehre, T. (2000, December 21). The history behind BYOND, DUNG and DanTom. *BYOND Developer Forum*. Retrieved March 7, 2011, from http://www.byond.com/developer/forum/?id=7050&display=1#7050

Hochet, Y. (2011). Jeux vidéo et enseignement de l'histoire et de la géographie. In S. Rufat & H. T. Minassian (Eds.), *Les jeux vidéos comme objet de recherche* (pp. 103-112). Questions Théoriques.

Howland, K. (2010, August 27). Flip: a visual and textual programming environment for teaching computation through game creation. University of Sussex. Retrieved from http://www.flipproject.org.uk/2010/08/flip-background-and-approach/

Howland, K., Good, J., & du Boulay, B. (2008). A Game Creation Tool which Supports the Development of Writing Skills: Interface Design Considerations. Presented at the Narrative and Interactive Learning Environments (NILE 2008), Edinburgh, United Kingdom.

Huizinga, J. (1951). *Homo ludens*. Gallimard.

Hunicke, R., Leblanc, M., & Zubek, R. (2004). MDA: A Formal Approach to Game Design and Game Research. In *Proceedings of the Challenges in Games AI Workshop, Nineteenth National Conference of Artificial Intelligence* (pp. 1--5). Presented at the Nineteenth National Conference of Artificial Intelligence, San Jose, USA. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.79.4561

Huynh-Kim-Bang, B. (2010, March). Design Patterns for Serious Games. Retrieved June 7, 2010, from http://seriousgames.lip6.fr/DesignPatterns/dp/publish

Huynh-Kim-Bang, B., & Labat, J. (2010). Design Patterns in Serious Games: how to mix fun and pedagogy? In *10th International Conference on Intelligent Tutoring Systems (ITS2010)*. Presented at the 10th International Conference on Intelligent Tutoring Systems (ITS2010), Pittsburg, USA.

Innovation & Next Practice Division. (2010). *The impact of web 2.0 technologies in the classroom - Kodu excerpt*. Melbourne, Australia: Department of Education and Early Childhood Development (Victoria State Government).

Jansiewicz, D. R. (1973). *The New Alexandria Simulation: A Serious Game of State and Local Politics*. Canfield Press.

Jansiewicz, D. R. (2011). The Game of Politics - Frequently Asked Questions. Retrieved February 8, 2011, from http://www.gameofpolitics.com/f__a__q_.htm

Järvinen, A. (2008, March 8). *Games without Frontiers: Theories and Methods for Game Studies and Design* (PhD Thesis). Finland: University of Tampere. Retrieved from http://acta.uta.fi/english/teos.php?id=11046

Jenkins, H., Camper, B., Chisholm, A., Grigsby, N., Klopfer, E., Osterweil, S., Perry, J., et al. (2009). From Serious Games to Serious Gaming. In U. Ritterfeld, M. Cody, & P. Vorderer (Eds.), *Serious Games: Mechanisms and Effects* (1er ed.). Routledge.

Johnson, A. (1999). The first 'Official' Castle Smurfenstein Home Page. Retrieved May 21, 2010, from http://www.evl.uic.edu/aej/smurf.html

Juul, J. (2003). The game, the player, the world: looking for a heart of gameness. In C. Marinka & R. Joost (Eds.), *Level Up Conference Proceedings: Proceedings of the 2003 Digital Games Research Association Conference* (pp. 30-45). Utrecht: University of Utrecht. Retrieved from http://www.digra.org/dl/display_html?chid=05163.50560.pdf

Juul, J. (2005). *Half-real : video games between real rules and fictional worlds*. Cambridge Mass.: MIT Press.

Juul, J. (2009). *A Casual Revolution*. MIT Press.

Kafai, Y. B. (1994). *Minds in Play: Computer Game Design As A Context for Children's Learning*. Routledge.

Kafai, Y. B. (2006). Playing and Making Games for Learning. *Games and Culture*, *1*(1), 36 -40. doi:10.1177/1555412005281767

Kahn, M. A., & Perez, K. M. (2009). The Game of Politics Simulation: An Exploratory Study. *Journal of Political Science Education*, *5*(4), 332. doi:10.1080/15512160903253707

Kallmann, M., & Thalmann, D. (1999). Direct 3D interaction with smart objects. In *Proceedings of the ACM symposium on Virtual reality software and technology - VRST '99* (pp. 124-130). Presented at the the ACM symposium, London, United Kingdom. doi:10.1145/323663.323683

Kapalka, J. (2006). 10 Ways to Make a BAD Casual Game. *Casual Connect Magazine*. Retrieved November 3, 2010, from http://www.casualconnect.org/content/gamedesign/kapalka-tenways.html

Kelley, D. (1988). *The Art of Reasoning* (1er ed.). Norton & Company.

Kellner, C. (2000, December 22). *La médiation par le cédérom « ludo-éducatif ». Approche communicationnelle*. Metz, France: Université de Metz.

Kellner, C. (2007). *Les cédéroms pour jouer ou pour apprendre ?* L'Harmattan.

Kent, S. L. (2001). *The Ultimate History of Video Games* (1er ed.). Three Rivers Press.

King, B., & Borland, J. (2003). *Dungeons and Dreamers: The Rise of Computer Game Culture from Geek to Chic* (1er ed.). McGraw-Hill Osborne Media.

Knopf, J., & Ford, N. (2000). The History of Shareware. *Association of Software Professionals*. Retrieved January 9, 2011, from http://www.asp-software.org/users/history-of-shareware.asp

Knorr, A. (2007). Game Modding. unpublished article. Retrieved from xirdal.lmu.de/downloads/KNORR_2007_Game_modding.pdf

Kohler, C. (2004). *Power-Up: How Japanese Video Games Gave the World an Extra Life*. BRADY GAMES.

Koster, R. (2004). *A Theory of Fun for Game Design* (1er ed.). Paraglyph Press.

Koster, R. (2005, March). *A Grammar of Gameplay - Game Atoms: can games be diagrammed?* Presented at the Game Developers Conference 2005, San Fransisco, USA.

Koster, R. (2009, September 18). *Games Are Math: 10 Core Mechanics That Drive Compelling Gameplay*. Presented at the Game Developer Conference Austin. Retrieved from http://www.raphkoster.com/gaming/gdca2009.shtml

Kremer, M. (2009). Creating Games For The Casual Audience.

Kremers, R. (2009). *Level Design: Concept, Theory, and Practice*. AK Peters.

Krupa, F. (2003, February 5). Virtools SDK Architecture and Principles.

Kunze, F., & Giola, G. (2008, August 1). A Brief Overview of Eamon. *Eamon Adventurer's Guild Online*. Retrieved December 11, 2010, from http://www.eamonag.org/pages/eamon_overview.htm

Kushner, D. (2002, July 2). The Mod Squad. *Popular Science*. Retrieved from http://www.popsci.com/gear-gadgets/article/2002-07/mod-squad

Kushner, D. (2004). *Masters of Doom: How Two Guys Created an Empire and Transformed Pop Culture*. Random House Trade Paperbacks.

Laukkanen, T. (2005). *Modding scenes: Introduction to user-created content in computer gaming*. Hypermedia Laboratory Net Series 9. Tampere, Finland: University of Tampere.

Lavigne, M. (2010). Interactivité, interactions et intégration scolaire. Presented at the Ludovia 2010, Ax-Les-Thermes, France.

Le Diberder, A., & Le Diberder, F. (1993). *Qui a peur des jeux video ?* La Découverte.

Le Roy, B. (2006, April 1). Gaming 2.0. Retrieved May 21, 2010, from http://weblogs.asp.net/bleroy/archive/2006/04/01/Gaming-2.0.aspx

Leblanc, M. (2005). Tools for creating Dramatic games Dynamics (2005). In K. Salen & E. Zimmerman (Eds.), *The Game Design Reader: A Rules of Play Anthology*. The MIT Press.

Lecky-Thompson, G. W. (2007). *Video Game Design Revealed* (1er ed.). Cengage Learning.

Lee, S. (2003). I Lose, Therefore I Think: A Search for Contemplation amid Wars of Push-Button Glare. *Game Studies*, *3*(2). Retrieved from http://www.gamestudies.org/0302/lee/

Letourneux, M. (2006). La question du genre dans les jeux vidéo. In S. Genvo (Ed.), *Le game design de jeux vidéo : Approches de l'expression vidéoludique*. Editions L'Harmattan.

Levieux, G. (2011, May 9). *Mesurer la difficulté dans les jeux vidéo* (PhD Thesis). Paris, France: Conservatoire National des Arts et Métiers. Retrieved from http://guillaumelevieux.com/thesisManuscript.php

Lhuillier, B. (2011). *Concevoir un serious game pour un dispositif de formation*. FYP EDITIONS.

Lieberman, D. A. (2001). Management of chronic pediatric diseases with interactive health games: theory and research findings. *The Journal of Ambulatory Care Management*, *24*(1), 26-38.

Linderoth, J. (2010). Why gamers don't learn more - An ecological approach to games as learning environments. In L. Petri, T. A. Mette, V. Harko, & W. Annika (Eds.), *Proceedings of DiGRA Nordic 2010: Experiencing Games: Games, Play, and Players*. Stockholm: University of Stockholm. Retrieved from http://www.digra.org:8080/Plone/dl/display_html?chid=10343.51199.pdf

Llanas, J. (2009). *Pedagame: Jeux vidéo, éducation et formation*. Paris, France: Pedagame.

Love, N., Hinrichs, T., Genesereth, M., & Skufza, E. (2008). *General Game Playing: Game Description Language Specification*. USA: Stanford University. Retrieved from http://games.stanford.edu/language/language.html

Malone, T. W. (1981). Toward a theory of intrinsically motivating instruction. *Cognitive Science*, *5*(4), 333-369.

Manning, J. (2004). *The Emblem*. Reaktion Books.

Marfisi-Schottman, I., George, S., & Tarpin-Bernard, F. (2010). Tools and Methods for Efficiently Designing Serious Games. In *Games Based Learning, ECGBL 2010* (pp. 226-234).

Marfisi-Schottman, I., Sghaier, A., George, S., Tarpin-Bernard, F., & Prévôt, P. (2009). Towards Industrialized Conception and Production of Serious Games. Presented at the Internantial Conference on Technology and Education (ICTE), Paris, France. Retrieved from http://arxiv.org/abs/0911.4262

Maurin, F. (2010, July 15). Jeu vidéo : "Je perds donc je pense". *LeMonde.fr*. Retrieved May 18, 2011, from http://www.lemonde.fr/actualite-medias/article/2010/07/15/jeu-video-je-perds-donc-je-pense-serious-games-2-5_1385386_3236.html

McMahon, M. (2009). Using the DODDEL model to teach serious game design to novice designers. Presented at the ASCILITE 2009, Auckland, New Zealand.

Meigs, T. (2003). *Ultimate Game Design: Building Game Worlds* (1er ed.). McGraw-Hill Osborne Media.

Meloni, W. (2010). *THE BRIEF: 2009 Ups and Downs in video game industry*. M2 Research. Retrieved from http://www.m2research.com/the-brief-2009-ups-and-downs.htm

Michael, D., & Chen, S. (2005). *Serious Games: Games That Educate, Train, and Inform* (1er ed.). Course Technology PTR.

Minter, J., & Minter, P. (2009). *A History of Llamasoft*. Llamasoft.

Molenda, M. (2003). In Search of the Elusive ADDIE Model. *Performance Improvement*, *42*(5). Retrieved from http://www.eric.ed.gov/ERICWebPortal/detail?accno=EJ676523

Moreno-Ger, P., Martínez-Ortiz, I., Sierra, J. L., & Fernández-Manjón, B. (2008). A Content-Centric Development Process Model. *Computer*, *41*(3), 24-30.

Moskal, B., Lurie, D., & Cooper, S. (2004). Evaluating the effectiveness of a new instructional approach. In *ACM SIGCSE Bulletin* (Vol. 36, pp. 75–79). New York, NY, USA: ACM. doi:10.1145/1028174.971328

Muratet, M. (2010, December 2). *Conception, réalisation et évaluation d'un jeu sérieux de stratégie temps réel pour l'apprentissage des fondamentaux de la programmation* (PhD Thesis). Toulouse, France: Université de Toulouse.

Myers, D. (1990). Computer game genres. *Play & Culture*, *3*(1990).

Nations, D. (2008, December 1). Web 2.0 + Gaming = Spore? *About.com : Web Trends*. Retrieved March 2, 2011, from http://webtrends.about.com/b/2008/12/01/web-20-gaming-spore.htm

Natkin, S. (2006). *Video Games and Interactive Media: A Glimpse at New Digital Entertainment* (illustrated edition.). A K Peters.

Neubauer, S. (2006, May). *The Gameplay Video Segmentation Method: A heuristic method for gameplay analysis using video segmentation* (Master thesis). Stuttgart, Germany: Stuttgart Media University.

Nieborg, D. B., & van der Graaf, S. (2008). The mod industries? The industrial logic of non-market game production. *European Journal of Cultural Studies*, *11*(2), 177 -195. doi:10.1177/1367549407088331

Onlineformapro. (2010, August). Plaquette de présentation de SGtools. Onlineformapro. Retrieved from http://www.onlineformapro.com/serious_game.php

O'Reilly, T. (2005). What is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software. Presented at the Web 2.0 Conference, San Fransisco. Retrieved from http://oreilly.com/web2/archive/what-is-web-20.html

Overmars, M. (2004). Teaching computer science through game design. *Computer*, *37*(4), 81-83. doi:10.1109/MC.2004.1297314

Papert, S. A. (1993). *Mindstorms: Children, Computers, And Powerful Ideas* (2 ed.). Basic Books.

Papert, S. A., & Harel, I. (1991). Situating Constructionism. In S. Papert & I. Harel (Eds.), *Constructionism*. Ablex Publishing.

Pardew, L., Pugh, S., Nunamaker, E., Iverson, B. L., & Wolfley, R. (2004). *Game Design for Teens* (1er ed.). Course Technology PTR.

Parlett, D. (1999). *Oxford History of Board Games* (illustrated edition.). Oxford University Press, USA.

Pausch, R., Brunette, T., Capehart, A., Conway, M., Cosgrove, D., DeLine, R., Durbin, J., et al. (1995). A Brief Architectural Overview of Alice, a Rapid Prototyping System for Virtual Reality. *IEEE Computer Graphics and Applications*, *15*(3), 8-11.

Pedersen, R. E. (2003). *Game Design Foundations.* Wordware Publishing, Inc.

Peppler, K. A., & Kafai, Y. B. (2007). From SuperGoo to Scratch: exploring creative digital media production in informal learning. *Learning, Media and Technology*, *32*(2), 149-166.

Perry, D., & DeMaria, R. (2009). *David Perry on Game Design: A Brainstorming Toolbox*. Charles River Media.

Pierce, S. (2011, June 21). Towards a Rule-Based Game Engine. *Gamasutra*. Retrieved June 26, 2011, from http://www.gamasutra.com/blogs/ShayPierce/20110621/7830/Towards_a_RuleBased_Game_Engine.php

Pirou, J. (2010, September). Faites vos jeux - Rencontre avec François Lionet, cofondateur de la Clickteam. *IG Magazine*, (10), 174-179.

Pivec, M. (2010). *Engage Catalogue of Games for Learning 2009-2010*. ENGAGE learning. Retrieved from http://www.engagelearning.eu/?p=994

Postigo, H. (2007). Of Mods and Modders: Chasing Down the Value of Fan-Based Digital Game Modifications. *Games and Culture*, *2*(4), 300 -313. doi:10.1177/1555412007307955

Prensky, M. (2007). *Digital Game-based Learning* (Paragon House Ed.). Paragon House Publishers.

Rankin, J. R., Vargas, S. S., & Taylor, P. F. (2009). Testing metaphorical educational FPS games. *Int. J. Comput. Games Technol.*, *2009*, 1-5.

Rieber, L. P., Luke, N., & Smith, J. (1998). Projet KID DESIGNER: Constructivism at Work through Play. *Meridian*, *1*(1).

Robertson, J., & Howells, C. (2008). Computer game design: Opportunities for successful learning. *Computers & Education*, *50*(2), 559-578.

Robertson, J., & Nicholson, K. (2007). Adventure Author: a learning environment to support creative design. Presented at the Adventure Author: a learning environment to support creative design, Aalborg, Denmark.

Rogers, S. (2010). *Level Up!: The Guide to Great Video Game Design*. Wiley.

Rohrl, D. (2008). *2008- 2009 Casual Games White Paper*. USA: IGDA.

Rollings, A., & Adams, E. (2003). *Andrew Rollings and Ernest Adams on Game Design* (Ltd Rmst.). New Riders Games.

Rollings, A., & Morris, D. (2003). *Game Architecture and Design: A New Edition*. New Riders Games.

Rouchier, J., & Retaux, X. (2002). Realism vs Surprise and Coherence: Different Aspect of Playability in Computer Games.

Rouet, F. (2009). *La création dans l'industrie du jeu vidéo*. Paris, France: Ministère de la Culture et de la Communication.

Rouse, R. (2000, March 23). Designing Design Tools. Retrieved June 16, 2010, from http://www.gamasutra.com/view/feature/3443/designing_design_tools.php?print=1

Rouse, R. (2001). *Game Design: Theory and Practice*. Wordware Publishing, Inc.

Roy, P. V., & Haridi, S. (2004). *Concepts, Techniques, and Models of Computer Programming*. MIT Press. Retrieved from http://www.info.ucl.ac.be/~pvr/book.html

Rufat, S., & Minassian, H. T. (Eds.). (2011). *Les jeux vidéos comme objet de recherche*. Questions Théoriques.

Salen, K. (2009). *Gamestar Mechanic Learning Guide*. Institute of Play. Retrieved from http://www.gamestarmechanic.com/

Salen, K., & Zimmerman, E. (2003). *Rules of play*. MIT Press.

Salen, K., & Zimmerman, E. (Eds.). (2005). *The Game Design Reader: A Rules of Play Anthology*. The MIT Press.

Saltzman, M. (1999). *Game Design: Secrets of the Sages*. BRADY GAMES.

Sauvé, L. (2010a). Introduction au guide. In L. Sauvé & D. Kaufman (Eds.), *Jeux et simulations éducatifs : Etudes de cas et leçons apprises* (pp. 463-465). Presses de l'Université du Québec.

Sauvé, L. (2010b). La conception d'une coquille générique de jeux éducatifs. In L. Sauvé & D. Kaufman (Eds.), *Jeux et simulations éducatifs : Etudes de cas et leçons apprises* (pp. 493-528). Presses de l'Université du Québec.

Sauvé, L. (2010c). La validation d'une coquille générique de jeux éducatifs. In L. Sauvé & D. Kaufman (Eds.), *Jeux et simulations éducatifs : Etudes de cas et leçons apprises* (pp. 546-563). Presses de l'Université du Québec.

Sauvé, L., & Kaufman, D. (Eds.). (2010). *Jeux et simulations éducatifs : Etudes de cas et leçons apprises*. Presses de l'Université du Québec.

Sawyer, B. (2007). The "Serious Games" Landscape. Presented at the Instructional & Research Technology Symposium for Arts, Humanities and Social Sciences, Camden, USA.

Sawyer, B. (2009). Foreword : From Virtual U to Serious Game to Something Bigger. In U. Ritterfeld, M. Cody, & P. Vorderer (Eds.), *Serious Games: Mechanisms and Effects* (1er ed.). Routledge.

Sawyer, B., & Rejeski, D. (2002). *Serious Games: Improving Public Policy Through Game-based Learning and Simulation*. Woodrow Wilson International Center for Scholars.

Sawyer, B., & Smith, P. (2008). Serious Games Taxonomy. Presented at the Serious Games Summit 2008, San Fransisco, USA.

Schell, J. (2008a). *The Art of Game Design: A Deck of Lenses* (1er ed.). Schell Games.

Schell, J. (2008b). *The Art of Game Design: A Book of Lenses*. Morgan Kaufmann.

Schrage, M. (1999). *Serious Play: How the World's Best Companies Simulate to Innovate* (1er ed.). Harvard Business Press.

Shaffer, D. W. (2007). *How Computer Games Help Children Learn* (Reprint.). Palgrave Macmillan.

Sicart, M. (2009). Newsgames: Theory and Design. In *Proceedings of the 7th International Conference on Entertainment Computing* (pp. 27-33). Pittsburgh, PA: Springer-Verlag. Retrieved from http://portal.acm.org/citation.cfm?id=1483460.1483465&coll=GUIDE&dl=GUIDE

Sihvonen, T. (2009, May 23). *Players Unleashed! Modding The Sims and the Culture of Gaming* (PhD Thesis). Turku, Finland: University of Turku. Retrieved from https://www.doria.fi/handle/10024/44913

Smillie, K. (2010). *Some Topics in Computing*. Keith Smillie. Retrieved from http://webdocs.cs.ualberta.ca/~smillie/Topics/Topics.html

Smith, A. M., Nelson, M. J., & Mateas, M. (2010). Ludocore: A Logical Game Engine for Modeling Videogames. In *IEEE Conference on Computational Intelligence and Games (CIG)*. University of Copenhagen, Denmark.

Smith, D. C., Cypher, A., & Spohrer, J. (1994). KidSim: programming agents without a programming language. *Commun. ACM*, *37*(7), 54-67. doi:10.1145/176789.176795

Smith, J. D. (2009, August 12). *Raptor: Sketching Video Games With a Tabletop Computer* (PhD Thesis). Kingston, Canada: Queen's University.

Smith, J. D., & Graham, N. (2010). Raptor: Sketching Games with a Tabletop Computer. In *Future Play*. Presented at the Future Play, Vancouver, Canada.

Smith, R. (2008). *Rogue Leaders: The Story of LucasArts*. Chronicle Books.

SNJV. (2009). *Enquête sur la formation initiale dans l'industrie du jeux vidéo en France*. France: SNJV.

Soderberg, H. (2001). *The Serious Game* (1er ed.). Marion Boyars Publishers Ltd.

Sotamaa, O. (2009, April 25). *The Player's Game - Towards Understanding Player Production Among Computer Game Culture* (Master thesis). Tampere, Finland: University of Tampere.

Sotamaa, O. (2010). Play, Create, Share? Console Gaming, Player Production and Agency. *The Fibreculture Journal*, *2010*(16). Retrieved from http://sixteen.fibreculturejournal.org/play-create-share-console-gaming-player-production-and-agency/

Stolee, K. (2010). *Kodu Language and Grammar Specification*. Microsoft Research. Retrieved from http://research.microsoft.com/en-us/projects/kodu/

Stora, M., & de Dinechin, B. (2005). *Guérir par le virtuel : Une nouvelle approche thérapeutique*. Presses de la Renaissance.

St-Pierre, R. (2006, January). *La conception de jeux vidéos éducatifs : une méthodologie de recherche/création* (PhD Thesis). Montréal, Canada: Université du Québec. Retrieved from http://www.clikmedia.ca/CM/

Strange Agency. (2006). Strange Analyst Manual. Strange Agency.

Suits, B. (2005). Construction of a definition (1990). In K. Salen & E. Zimmerman (Eds.), *The Game Design Reader: A Rules of Play Anthology*. The MIT Press.

Tajè, P. (2007, July 27). Gameplay Deconstruction: Elements and Layers. *Game Career Guide*. Retrieved from http://www.gamecareerguide.com/features/355/gameplay_deconstruction_elements_.php

Taylor, T. (2009). The Assemblage of Play. *Games and Culture*, *4*(4), 331 -339. doi:10.1177/1555412009343576

Thomas, P. (2010). Le suivi de l'apprenant dans un serious game intégré au sein d'une plateforme d'apprentissage. Presented at the Troisièmes Rencontres Jeunes Chercheurs en EIAH, Université Lyon I, Lyon.

Tinsman, B. (2008). *The Game Inventor's Guidebook: How to Invent and Sell Board Games, Card Games, Role-playing Games & Everything in Between!* Morgan James Publishing.

Tolino, A. (2008, December). *Gaming 2.0 - Computer Games and Cultural Production - Participation Analysis of Computer Gamers in a Convergent Media Culture and Taxonomy of Ludic Artifacts* (PhD Thesis). Vienna, Austria: University of Applied Arts.

Tolino, A. (2009, May 14). Beyond Play: Analyzing Player-Generated Creations. Retrieved August 30, 2010, from http://www.gamasutra.com/view/feature/4008/beyond_play_analyzing_.php?print=1

Torrente, J., Del Blanco, A., Marchiori, E. J., Moreno-Ger, P., & Andndez-Manjó Andn, B. (2010). <e-Adventure>: Introducing educational games in the learning process (pp. 1121-1126). Presented at the IEEE EDUCON 2010, Madrid, Spain: IEEE. Retrieved from http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5493056

Torrente, J., Marchiori, E., & del Blanco, A. (2008). The <e –Adventure> Platform: Editor User Manual. Retrieved from http://e-adventure.e-ucm.es/tutorial/

Torrente, J., Moreno-Ger, P., Fernández-Manjón, B., & Blanco, Á. (2009). Game-Like Simulations for Online Adaptive Learning: A Case Study. In *Proceedings of the 4th International Conference on E-Learning and Games: Learning by Playing. Game-based Education System Design and Development*, Edutainment '09 (pp. 162–173). Berlin, Heidelberg: Springer-Verlag. doi:http://dx.doi.org/10.1007/978-3-642-03364-3_21

Trefry, G. (2010). *Casual Game Design: Designing Play for the Gamer in ALL of Us* (1er ed.). Morgan Kaufmann.

Vandamme, E. (2010, January 1). Serious Game Journalistique. *Pôle numérique de l'ESJ Lille*. Retrieved April 1, 2011, from http://numerique.esj-lille.net/2010/01/01/serious-game-journalistique/

Vialade, Q. (2011). *Rapport de Stage : Ludoscience*. Castres, France: DUT Service & Réseaux de communication.

Vick, E. H., McDaniel, R., & Jacobs, S. (2010). Using Semiotic Grammars for the Rapid Design of Evolving Video Game Mechanics. Presented at the SIGGRAPH 2010, Los Angeles.

Wagner, J. A. (2002, April 16). Triumph of the mod. Retrieved August 30, 2010, from http://www.salon.com/technology/feature/2002/04/16/modding/print.html

Wolf, M. J. P. (2002). *The Medium of the Video Game* (1er ed.). University of Texas Press.

Wolf, M. J. (2007). *The Video Game Explosion: A History from Pong to Playstation and Beyond*. Greenwood Press.

Yessad, A., Labat, J., & Kermorvant, F. (2010). SeGAE: A Serious Game Authoring Environment. Presented at the The 10th IEEE International Conference on Advanced Learning Technologies (ICALT 2010), Sousse, Tunisia.

Yucel, I., Zupko, J., & El-Nasr, M. S. (2006). IT education, girls and game modding. *Interactive Technology and Smart Education*, *3*(2), 143-156. doi:10.1108/17415650680000059

Zagal, J. P. (2010). *Ludoliteracy: Defining, Understanding, and Supporting Games Education* (1er ed.). ETC Press.

Zagal, J. P., & Bruckman, A. (2008). Novices, Gamers, and Scholars: Exploring the Challenges of Teaching About Games. *Game Studies*, *8*(2).

Zyda, M. (2005). From Visual Simulation to Virtual Reality to Games. *Computer*, *38*(9), 25-32.