Learning game 2.0 : Support à l'Apprentissage Collaboratif par la Modification de Jeux Vidéo

Baptiste Monterrat, Elise Lavoué, Sébastien George

Learning game 2.0 : Support à l'Apprentissage Collaboratif par la Modification de Jeux Vidéo

▶ Baptiste MONTERRAT (INSA-Lyon, LIRIS), Élise LAVOUÉ (Université Jean Moulin Lyon 3, MAGELLAN, LIRIS), Sébastien GEORGE (INSA-Lyon, LIRIS)

■ RÉSUMÉ • Les outils de *modding* créés depuis quelques années permettent aux joueurs de modifier leurs jeux vidéo de plus en plus facilement. On observe que les personnes qui ont modifié un jeu acquièrent à cette occasion des connaissances sur son fonctionnement et ses contenus. L'idée que nous développons ici est l'application de ce principe aux *learning games*, pour faire du jeu 2.0 un moyen d'appropriation des connaissances accessible à tout type de public. Dans cet article, nous proposons un modèle d'activité éducative et collaborative basée sur le *modding*, et une architecture générique de système informatique supportant une telle activité. Nous avons réalisé un prototype implémentant cette architecture, et une expérimentation pour tester le modèle d'activité proposé. Les résultats démontrent que les outils collaboratifs conçus tendent à favoriser l'apprentissage.

■ MOTS-CLÉS • *modding*, *learning game*[1], apprentissage collaboratif.

1. Introduction

Pendant les dix dernières années, de nouvelles formes d'outils ont émergé dans le domaine du jeu vidéo, dans le but de permettre aux développeurs en herbe de créer leurs propres jeux. Les éditeurs de jeux et les usines à jeux comptent parmi ces outils. Nous pouvons les appeler des GDK (« Game Development Kits »). Ils sont à l'origine de l'ère du « Jeu vidéo 2.0 ».

Parallèlement, les *learning games* font leurs preuves en matière d'éducation depuis plusieurs années. Avec le « *learning game 2.0* », nous présentons la modification de jeux éducatifs comme une nouvelle manière de s'approprier des connaissances. À la fin de leur étude, Djaouti *et al.* (2010) concluent que actuellement, « *les technologies du jeu 2.0 présentent des idées intéressantes rela-*

1 Nous souhaitons parler ici d'apprentissage au sens large : acquisition et construction de connaissances. Pour cette raison nous utilisons le terme de "learning game" plutôt que de "jeu éducatif", qui évoque généralement des jeux destinés aux enfants.

tives à la conception des jeux, mais ne semblent pas avoir assez pris conscience du potentiel « sérieux » des jeux vidéo ». Les travaux présentés ici visent à combler ce manque en proposant un *framework* informatique supportant des activités pédagogiques basées sur le *modding*. Ils reposent sur l'hypothèse que créer et modifier des contenus ludo-éducatifs en amène une connaissance plus profonde que le simple fait de jouer avec ces contenus.

Dans la partie 2, à travers une étude de l'état de l'art, nous expliquons les raisons qui nous ont amenés à choisir le *modding* comme activité pédagogique. Nous présentons ensuite en détail le *framework* et les outils qui accompagnent cette activité. Une expérimentation de notre proposition a été réalisée auprès de 16 personnes pour tester les éléments proposés. Nous en présentons la démarche et les résultats dans la section 4.

2. Apprentissage par le *modding*

2.1. Le potentiel du *modding* : état de l'art

Des utilisateurs devenant développeurs. Au début, la seule manière de modifier un jeu vidéo était d'accéder à son code source. Heureusement, l'apparition permanente de nouveaux types d'outils d'édition offre aux utilisateurs un accès plus facile au cœur des jeux. La manière la plus basique de modifier un jeu pour un utilisateur est d'accéder à ses paramètres. La plupart des jeux permettent ainsi de modifier par exemple le mode d'affichage ou le niveau de difficulté. Pendant les années 80, de nouveaux types d'outils sont arrivés : les éditeurs de niveaux. Ils ont contribué au succès de Lode Runner, l'un des premiers jeux à permettre aux utilisateurs de décrire eux-mêmes l'état initial d'un niveau (Djaouti, 2011). De nos jours, les joueurs ont enfin accès à de nombreux moteurs de jeux différents leur permettant de modifier complètement le comportement d'un jeu pour en construire un nouveau. Tavares et Roque (2007) ont démontré l'intérêt, pour un jeu, d'être conçu par un grand nombre de joueurs apportant chacun leurs idées, plutôt que par un petit nombre de professionnels. D'après Volk (2007), « *Le choix entre le rôle de concepteur et celui de joueur n'est pas binaire, puisque tous les niveaux d'implication peuvent être trouvés dans le monde du modding* ». Les jeux vidéo ayant besoin à la fois des idées des fans et du travail des concepteurs professionnels, les outils informatiques de conception doivent être adaptés à ces différents profils.

Des outils de *modding* de plus en plus accessibles. Plusieurs types d'outils ont été créés dans le but de simplifier la programmation. Ils visent tous à augmenter l'expressivité du mode de représentation tout en diminuant sa complexité. Le Web 2.0 a grandi notamment grâce aux éditeurs WYSIWYG (« *What You See*

Is What You Get »). Par analogie, le jeu 2.0 est en train de devenir accessible à tous grâce à cette nouvelle génération d'outils, fournissant aux utilisateurs une bonne représentation du jeu. Ils sont souvent composés de deux interfaces : l'une pour éditer l'état initial d'une scène ou d'un niveau de jeu, et l'autre pour en modifier le comportement. Le glisser-déposer d'objets depuis la librairie vers le jeu est un exemple d'outil ouvrant les éditeurs de niveau à un public plus large. Les interfaces offrant la possibilité de modifier le comportement d'un jeu deviennent elles aussi de plus en plus visuelles, comme le montrent ces exemples :

– *Stencyl* et *Flip* permettent à l'utilisateur de programmer un jeu sans écrire de code. Des blocs (*if*, *then*, *boolean*, etc.) et des fonctions existantes peuvent simplement être glissées, déposées et ordonnées dans une structure correspondant au comportement du jeu.

– *Warcraft III editor* va plus loin avec une représentation sous forme de déclencheurs (triggers), des règles composées d'un événement, de conditions et d'actions. Des notions de programmation comme les boucles et les fonctions sont alors rendues implicites, et n'ont pas besoin d'être maîtrisées par le développeur.

– *Kodu Game Lab* et *Game Develop* utilisent un moyen encore plus simple pour représenter le comportement d'un jeu: des règles composées uniquement de conditions et d'actions. Le moteur de jeu agit alors comme si les conditions étaient testées en permanence, et les actions correspondantes déclenchées le cas échéant. Ce système est tellement simple que *Kodu* permet à des enfants de concevoir leurs propres jeux. Aussi, *Kodu* permet un accès à l'éditeur de comportement depuis l'intérieur du jeu lui-même : sélectionner un objet mène directement à la liste des règles qui définissent le comportement de cet objet.

Tous ces éditeurs ne sont que quelques exemples parmi ceux développés récemment. Pour plus de détails, Djaouti *et al*. (2010) ont présenté une étude de 15 outils de « *gaming 2.0* », et de la manière dont ils supportent la conception de jeux sérieux.

Le besoin d'outils libres. Volk (2007) a été l'un des premiers à parler de « *game development 2.0* ». Il a observé que les consommateurs ne participent pas aussi librement qu'on peut l'imaginer : « *Ils sont encadrés par les entreprises dont le but est de réaliser des études de marché* ». Plusieurs auteurs nous mettent en garde contre ce phénomène. Sotamaa (2005) décrit les compétitions de *modding* comme un moyen d'encadrer le travail gratuit réalisé par les *modders*. Des affordances dans les outils de développement ont été mises en évidence par Scacchi (2010). Elles « *servent à organiser et diriger les actions des gens qui développent et partagent des mods [...] et profitent au studio de développement du jeu, à l'éditeur, et au revendeur* ». Enfin, Postigo (2008) a montré les limites du *modding* de jeux propriétaires, les *modders* n'ayant pas le droit de partager leurs créations. Toutes

ces observations nous amènent à préférer l'utilisation de jeux et logiciels libres, comme un moyen de veiller à ce que des objectifs financiers ne viennent pas se substituer aux objectifs éducatifs.

Un travail efficace grâce aux communautés d'entraide. Loh et Byun (2009) ont modifié le jeu NeverWinter Nights en jeu sérieux, et ils témoignent de leur expérience. Ils ont ainsi créé Saving Adryanee, un jeu dont l'objectif *in-game* est de créer une potion de guérison, et dont l'objectif pédagogique est d'enseigner des notions de chimie. Ce travail complexe a été réalisé par une équipe de 4 personnes en seulement deux mois grâce à l'expérience de ceux qui ont réalisé des projets de *modding* avant eux. Ainsi, les développeurs expérimentés et les débutants sont organisés en communautés construites autour d'un jeu, ou d'un logiciel de conception. Ils communiquent grâce à des outils collaboratifs sur le web (chat, forums), ou parfois grâce à des outils intégrés au jeu lui-même (conversations entre avatars dans les jeux en ligne). C'est l'association des outils collaboratifs et de l'esprit d'entraide des *modders* qui permet aux débutants de dépasser leurs difficultés rapidement.

Le *modding* comme solution pour apprendre à programmer. Des étudiants ont pu créer des *mods* par eux-mêmes dans diverses études. McAtamney (2005) les a fait utiliser Crytek engine pour modéliser le futur campus de leur université. À travers cette expérience, ils ont appris à utiliser le C++, direct X, le langage de script LUA, tout en développant leurs compétences en mathématiques, en physique, en conception 3D et en programmation événementielle. Dans une activité plus amusante, El-Nasr et Smith (2006) ont montré qu'il existe des outils de *modding* adaptés à différents types d'apprenants. Ils ont expérimenté l'éditeur de Warcraft III avec des lycéens, leur permettant de créer un jeu en seulement trois jours, afin d'apprendre les bases de l'algorithmique. Ils ont ensuite utilisé Web Driver et Unreal Engine 2.5 avec des étudiants en informatique. En plus des compétences techniques acquises, ces expériences ont permis aux étudiants de se familiariser avec le processus de développement logiciel. En effet, les étapes de développement d'un *mod* (élaboration du cahier des charges, conception, implémentation, test) et les étapes de développement d'un logiciel sont sensiblement les mêmes (Cignoni, 2001).

2.2. Proposition d'une activité basée sur le *modding*

Pour apprendre le contenu du jeu. À travers toutes ces études, le *modding* a principalement été utilisé pour enseigner l'informatique, les mathématiques, ou pour développer les capacités de travail en équipe. Nous pensons que le *modding* peut être utilisé pour apprendre n'importe quel domaine. Ainsi, nous proposons maintenant de réfléchir à une activité qui semble avoir été peu étu-

diée : l'apprentissage du contenu d'un jeu éducatif par le *modding* (Monterrat *et al*., 2012). En effet, si les *learning games* aident à acquérir des savoirs contenus dans le jeu, on peut supposer que le fait de modifier ces jeux permet de se les approprier de manière plus approfondie. Oblinger (2006) explique que le potentiel éducatif d'un *learning game* dépend du niveau d'implication du joueur. Notre but est ici d'impliquer le joueur au maximum en lui permettant de modifier le jeu. Un jeu vidéo éducatif modifiable fournirait ainsi à l'utilisateur une nouvelle manière d'apprendre par le jeu. Cette méthode s'apparente à l'approche constructiviste de l'apprentissage, en offrant à l'utilisateur un moyen de reconstituer lui-même les connaissances pour les acquérir (Lave et Wenger, 1991).

Moshirnia (2007) a étudié le *modding* comme un moyen pour les professeurs de créer des *learning games*. Ils ont enseigné la révolution américaine grâce à un *mod* du jeu Civilisation IV. En effet, pour l'apprentissage basé sur le jeu, la conception du jeu fait partie du travail du professeur, car il est celui qui sait comment transmettre au mieux son savoir. Par notre proposition d'activité, nous souhaitons permettre à l'étudiant de jouer le rôle du professeur. Pour cause, concevoir un jeu éducatif est une manière d'enseigner, car le jeu est un outil de transmission des connaissances. Aussi, enseigner est une manière d'apprendre. La transmission des savoirs impliquant un niveau de compréhension plus profond, elle est un vecteur d'apprentissage. Nous proposons ainsi à l'apprenant de jouer le rôle de l'enseignant pour atteindre un autre niveau d'expertise.

Une activité ludique à la portée de tous. Nous voulons que cette activité éducative soit accessible à tous, avec ou sans compétences en programmation. Pour cela nous nous tournons alors vers les technologies présentées en partie 2.1.2. Aussi nous pensons que la conception d'un jeu à partir de zéro est une épreuve trop difficile, laissant l'utilisateur face à de nombreuses questions techniques qui ne répondent pas aux objectifs pédagogiques. Si on utilise le *modding*, une structure de jeu a déjà été créée par le professeur. Elle permet aux apprenants de se focaliser exclusivement sur les contenus du jeu. Imaginez par exemple un jeu conçu pour apprendre les langues, où le joueur doit déplacer son avatar, et aller discuter avec des personnages pour qu'ils lui confient des quêtes dans la langue à apprendre. Si l'apprenant devait créer le jeu dans son ensemble, il perdrait beaucoup de temps à la création des personnages, de l'avatar et de la manière d'animer ses déplacements. Avec le *modding*, tout cela existe déjà dans le jeu. L'élève peut donc se concentrer sur l'ajout de nouveaux dialogues, ceci étant un moyen direct d'apprendre la langue.

De plus avec le *modding*, l'apprenant est amené à chercher de nouvelles informations par lui-même, si bien qu'il peut acquérir des notions qui vont au-delà

du savoir du professeur. D'après le modèle Magic Bullet (Becker, 2011), les apprenants vont ainsi augmenter la part de « l'apprentissage externe » (ce que les apprenants vont chercher hors du jeu), et l'inclure d'eux même dans « l'apprentissage obligatoire » (les savoirs contenus dans le jeu qu'il est nécessaire de maîtriser pour gagner). Les évolutions du jeu profiteront alors aux futurs joueurs, lors d'une nouvelle session d'apprentissage. Cette méthode de développement permet aussi au savoir d'être évolutif, dans le but d'être toujours à jour, à l'image d'un wiki que ses utilisateurs maintiennent fidèle au monde actuel qui évolue constamment. Un jeu ainsi modifiable pourrait également être facilement adapté à plusieurs types de publics, en étant modifié par des personnes de ce public elles-mêmes.

Une activité motivante. Sotamaa (2008) présente les différentes sources de motivation des *modders* : jouer avec l'éditeur de jeu, chercher, coopérer et faire de l'expression artistique. Dans une activité pédagogique basée sur le *modding*, nous pensons aussi au plaisir de créer quelque-chose de réutilisable directement. Nous proposons que chaque groupe d'apprenants présente le résultat de son travail aux autres groupes de la classe. Savoir que le jeu qu'ils sont en train de créer sera utilisé ensuite peut ainsi leur donner envie de produire un travail de qualité. De plus, de nombreux éléments d'esthétique sont souvent ignorés lors de la conception de jeux sérieux, car ils demandent un investissement conséquent et ne répondent pas à l'objectif principal, qui est éducatif. La négligence des éléments d'environnement amène souvent les jeux sérieux à être délaissés par les joueurs, au profit de jeux plus commerciaux. Ainsi, comme précisé précédemment, l'avantage du *game modding* est d'offrir des espaces dans lesquels un environnement de jeu attrayant est déjà présent.

Une activité collaborative. Selon Scacchi (2010, 2011) « *le modding est aussi un moyen d'apprendre à travailler avec les autres* ». Une personne qui crée un *mod* s'entraîne à travailler en équipe et à gérer un projet de groupe. Pendant les expériences de El-Nasr et Smith (2006), les étudiants ont dû d'abord se répartir les tâches au sein des binômes. Ils ont ensuite compris qu'il était bénéfique pour leur projet de communiquer avec les autres groupes. Ainsi quand le nombre de personnes impliquées dans un projet de *modding* croît, il y a certes plus de risques de voir apparaître des erreurs, mais il y a également plus de chances que les erreurs soient rapidement détectées et corrigées. Ang *et al*. (2005) ont étudié ce mécanisme d' « auto-régulation » dans les communautés de rédaction de wikis. Le même phénomène peut se produire avec les *learning games*, à condition qu'ils soient également conçus comme un média collaboratif. C'est justement ce que nous proposons ici : rendre accessible aux joueurs la modification des savoirs contenus dans un jeu, à l'image d'un lecteur de wiki qui souhaiterait parti-

ciper à sa rédaction. De la même façon, Backlund *et al.* (2011) proposent un outil de création de scénarii de jeu à des responsables d'intervention incidents n'ayant aucune compétence en conception de jeu pour les former à la gestion des incidents. L'activité reposant sur un mode itératif par lequel les scénarii sont construits, joués et reconstruits, aide, selon les auteurs, à la création d'un répertoire partagé entre les participants. De plus, l'outil supportant les discussions professionnelles soutient le développement d'une identité professionnelle.

3. Proposition d'un *framework* d'activité et d'outils adaptés

3.1. Un modèle simple et expressif

Le GDK est l'outil informatique associé à un jeu, permettant de modifier celui-ci. D'une part, il doit être assez simple pour une prise en main rapide, sans compétences de programmation requises. En effet, même si pour faire du *modding* il est nécessaire d'apprendre à « modder », nous souhaitons que cette étape soit aussi courte que possible afin que l'utilisateur puisse travailler sur les contenus du jeu dès le début. D'autre part, le GDK doit être assez riche et puissant pour permettre au *modder* de modifier la structure du jeu en profondeur et de définir des comportements complexes. Un système de règles composées de conditions et d'actions (tel que le propose Game Develop) nous semble être un bon compromis. Ces règles permettent de manipuler des structures particulières aux jeux que nous décrivons ci-après.

Les jeux sont généralement partagés en plusieurs espaces distincts, appelés niveaux, mondes ou cartes. Nous les appellerons des scènes, ce terme étant plus générique et moins associé à un type de jeu. D'autres éléments composent un jeu vidéo : textures, sons, personnages, objets, etc. Nous les désignerons tous par le terme « objet ». Ces éléments sont rassemblés sur le modèle de la figure 1.

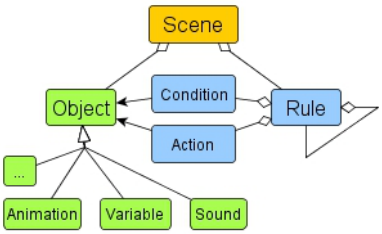

Figure 1 • Les éléments manipulés par le moteur de jeu et le GDK

Chaque scène d'un jeu est composée d'un ensemble d'objets et d'un ensemble de règles (pour décrire son comportement). Chaque type d'objet a des conditions et des actions qui lui sont associées. Voici quelques exemples :

– Condition pour un son : « Est ce que le son <musique de victoire> est joué en ce moment ? »

– Condition pour le clavier : « Est ce que la flèche gauche est pressée en ce moment ? »

– Action pour un son : « Jouer le son <explosion> »

– Action pour une image : « Déplacer l'image <Avatar> de 10 pixels vers la gauche »

Une règle peut être composée d'un nombre quelconque de conditions. S'il n'y en a aucune, l'action est déclenchée en permanence. Si une règle comporte plusieurs conditions, elles doivent être toutes vérifiées pour que l'action soit déclenchée, comme si elles étaient séparées par un ET logique. La manière la plus simple de réaliser un OU logique est alors de placer les conditions dans différentes règles. De plus, nous savons que toute formule logique peut être écrite sous la forme d'une disjonction de clauses conjonctives (forme normale disjonctive). Par conséquent, ce système de règles permet aux développeurs débutants de créer des conditions avancées sans pour autant avoir pris des cours de logique booléenne.

Si ce mode de représentation peut être compris très simplement, il peut aussi être lourd dans certains cas. En effet, la forme normale disjonctive n'est pas toujours le moyen de représentation le plus court d'une formule. Aussi nous proposons qu'une règle puisse être composée de sous-règles. Ce système est déjà en place dans Kodu et Game Develop. Les sous-règles d'une règle R ne sont déclenchées que quand les conditions de R ont déjà été vérifiées. Cela permet de ne pas réécrire les conditions de R dans chaque sous-règle. Une sous-règle étant une règle, ce modèle peut de nouveau s'appliquer récursivement. Par analogie avec le langage C, un ensemble de sous-règles peut être vu comme un bloc entre deux accolades. Ce modèle remplit alors les exigences fixées en termes de simplicité. C'est ensuite l'ensemble de conditions et d'actions fournies qui déterminera si les exigences sont remplies en termes d'expressivité.

Nous savons que lorsqu'un utilisateur fait des modifications (ajout d'objets, changement de règles...), il a besoin de tester régulièrement le jeu pour voir l'effectivité de ses changements. C'est pourquoi le processus de compilation pose un problème. Premièrement, des compilations répétées peuvent faire perdre beaucoup de temps au *modder*, en particulier si le jeu contient beaucoup d'éléments. Deuxièmement, ce processus implique que le joueur doit recommencer à jouer à son jeu depuis le début à chaque test. Le système sur lequel travaille le *modder* doit alors permettre des basculements rapides entre le GDK et le jeu. Nous voyons deux solutions possibles à ce problème : une compilation dyna-

mique ou l'utilisation d'un langage interprété, les deux permettant des changements pendant l'exécution du jeu.

3.2. Un modèle intégrant l'aspect collaboratif et éducatif

Comme nous l'avons expliqué dans la partie 2, le *modding* devient réellement intéressant éducativement et efficace quand il est fait collaborativement. Pour pouvoir travailler ensemble, les *modders* doivent dans un premier temps être capables de partager des idées et opinions, pour que le jeu émerge de choix collectifs. Dans un deuxième temps, ils ont également besoin de partager des éléments de jeu (cf. figure 1), pour que le jeu soit réellement le résultat d'un travail collectif. Une plate-forme sociale soutenant les discussions et le partage d'éléments semble alors nécessaire.

Dans notre modèle présenté en figure 2, nous intégrons des moyens de collaboration au modèle présenté en figure 1, et décrivons intégralement ce qu'est un jeu 2.0, en accord avec la définition donnée par Djaouti (2011) : « *toute application permettant à un utilisateur de créer, d'échanger et de jouer à un « contenu ludique »* ». Nous intégrons également à notre modèle les particularités du domaine éducatif, afin de concevoir un *learning game* 2.0. Parmi ces particularités se trouvent la poursuite d'objectifs éducatifs (et pas seulement ludiques), la nécessité d'évaluer et valider des savoirs et savoir-faires, et la présence d'un professeur. L'enseignant doit être le chef d'orchestre de l'activité d'apprentissage par le *modding*. Il est la personne expérimentée qui connaît les spécificités du domaine enseigné, qui sait prendre en compte les différents profils des élèves, gérer le temps à disposition, etc. Ses tâches sont :

- Concevoir un scénario d'activité en accord avec les objectifs pédagogiques.
- Créer le jeu de départ qui sera modifié par les joueurs.
- Gérer et accompagner l'activité pour aider les apprenants à sortir des impasses (techniques ou cognitives), et s'assurer qu'ils poursuivent les objectifs.
- Évaluer l'apprentissage et les apprenants.

Nous avons donc étendu le modèle précédent pour prendre en compte ces nouveaux aspects. La carte des connaissances élaborée par le professeur contient les connaissances du domaine à enseigner. Les éléments de cette carte sont reliés aux éléments de jeu qui permettent de les apprendre (liens G-K sur la figure 2). De l'autre côté, les éléments de connaissances sont également reliés à des discussions (liens K-D sur la figure 2). Les discussions contextuelles ont prouvé leur utilité dans des situations éducatives (George, 2004). Dans le cas d'une activité de *modding*, ces discussions peuvent être un espace d'échanges de savoirs entre apprenants. Elles peuvent aussi être un moyen pour le professeur de maintenir

le contact avec les étudiants pendant l'activité à distance, pour les recadrer ou les aider.

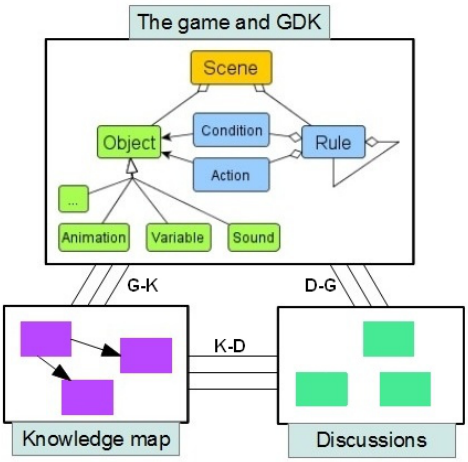

Figure 2 • Modèle de discussions contextuelles pour une activité d'apprentissage par le *modding*

Relativement au partage d'idées, il nous a également semblé important que ces discussions soient reliées à des éléments du jeu sur lequel les apprenants travaillent (liens D-G sur la figure 2). Ces liens aideront les étudiants à avoir des discussions structurées autour de l'élément de jeu sur lequel ils travaillent en parallèle. Voici deux exemples d'utilisation dans des contextes éducatifs différents :

– La connaissance « savoir appliquer la loi de la gravité » peut être reliée à un ensemble de règles simulant cette loi dans le jeu, et à une discussion entre élèves sur la gravité par rapport à la taille de la planète.

– La connaissance « savoir reconnaître un ornithorynque » peut être reliée à la scène d'un zoo et à une discussion sur les différences entre reptiles et mammifères.

Cette triple relation entre la structure du jeu modifié, la carte de connaissances et les discussions est au cœur de notre proposition d'environnement de *Learning Game* 2.0.

3.3. Des outils supportant les modifications collaboratives

Pour le projet de *modding* de leurs étudiants, McAtamney *et al*. (2005) avaient besoin de manipuler une base d'objets, un langage de script, et des outils de développement 3D. S'ils ont choisi d'utiliser Crytek engine, c'est en particulier

parce qu'il propose tous ces outils rassemblés dans un seul programme. Les outils basés sur notre modèle ont eux-aussi besoin d'être rassemblés au sein d'une application unique. Des changements fréquents entre l'interface du jeu, celle du GDK et celle de la plate-forme sociale seraient une barrière à l'apprentissage. Il nous semble donc important que toutes les facettes du *modding* fassent partie d'un environnement unique. La figure 3 présente notre proposition d'architecture pour une telle application.

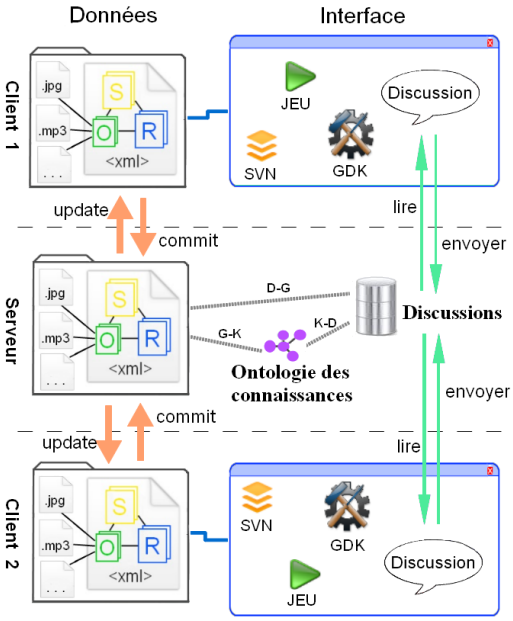

Figure 3 • Architecture d'un *learning game* 2.0

Pour que l'activité soit collaborative, il est indispensable que les éléments du jeu soient stockés sur un serveur. Chaque utilisateur (client) doit aussi disposer d'une version du jeu en local pour que chacun de ses tests n'affecte pas le projet collectif. Les scènes (S), objets (O) et règles (R) peuvent être enregistrées dans un fichier xml. Celui-ci doit contenir des liens vers les fichiers réels correspondant aux objets manipulés (images, sons, etc.). Il est ensuite facile de gérer cet ensemble de fichiers avec un système de gestion de versions inclus dans l'application.

Les deux commandes principales du SVN sont « *l'update* » (mettre à jour sa version locale du jeu) et le « *commit* » (appliquer nos modifications sur le projet collectif). Comme le développement de jeux implique la possibilité de faire des erreurs, le système doit donc également enregistrer sur le serveur les versions

précédentes du jeu et offrir un accès à toutes ces versions. Enfin, un système de gestion de versions peut aussi fusionner deux projets quand nécessaire, par exemple si deux utilisateurs ont fait des modifications en même temps.

L'application côté client doit faire cohabiter de nombreux outils. Chaque élément de jeu dispose de son fil de discussions associé (lien D-G). Pour l'implémentation de ce lien, nous proposons qu'un espace contenant les fils de discussion soit visible à côté du GDK. Ainsi, lorsqu'un apprenant sélectionne par exemple une scène pour travailler dessus, il voit en parallèle ce que ses collaborateurs ont proposé comme amélioration, la description de ce qu'ils ont déjà fait, ou encore des questionnements sur ce qu'ils ne comprennent pas encore.

L'enseignant doit avoir la possibilité de participer à ces discussions, car c'est un bon moyen d'accompagner les élèves dans leur travail. Les discussions doivent également être enregistrées sur le serveur. Une base de données sur le modèle relationnel semble être le format idéal. Concernant la carte des connaissances, l'ontologie semble être un format suffisamment souple et générique pour représenter des domaines variées, avec une sémantique plus ou moins riche en fonction des besoins de l'activité pédagogique.

Nous présentons une ébauche d'interface pour l'application client sur la figure 4. La liste des objets d'une scène (à gauche) se trouve à côté de l'éditeur de scènes (au centre), pour que les objets puissent être ajoutés à la scène par un simple glissé-déposé. L'espace central doit pouvoir laisser place à un éditeur de sons, d'images, de règles, ou autres, en fonction de l'objet sélectionné. Le contenu de la barre d'outils et la barre de discussion sont eux aussi fonction de l'objet sélectionné.

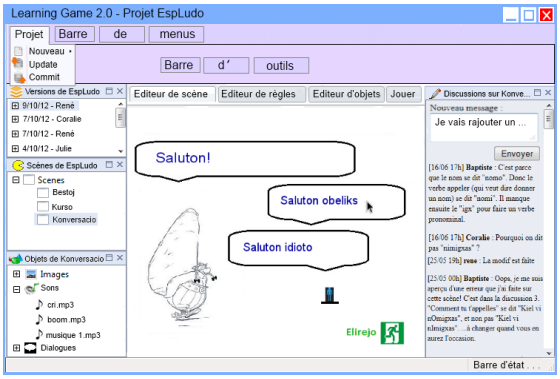

Figure 4 • Ébauche d'interface utilisateur pour le *learning game* 2.0

4. Implémentation et expérimentation

Il nous a semblé important d'expérimenter les différents éléments de notre proposition. Nous avons donc réalisé un prototype d'application basée sur le modèle *learning game* 2.0, et en avons testé différents aspects avec 2 groupes de volontaires. Les objectifs étaient :

– Tester la faisabilité technique du système proposé.

– Observer les avantages et inconvénients du *modding* en tant que méthode d'apprentissage.

– Savoir si les outils collaboratifs proposés facilitent l'apprentissage, et identifier leurs limites.

4.1. Démarche expérimentale

Déroulement de l'activité. Pour cette expérimentation, nous avons choisi d'enseigner l'Espéranto, et avons constitué deux groupes de 8 volontaires. Ces volontaires étaient majoritairement des étudiants (14) et deux retraités. Parmi les 16 participants, 7 avaient des compétences préalables en programmation informatique (cf. Tableau 1). Voici les différentes étapes de l'activité qui leur a été proposée. Les trois premières étapes ont eu lieu en présentiel, et les suivantes se sont faites à distance.

– 1. Pour commencer, un cours standard d'une heure sur les bases de la langue a été donné.

– 2. Ils ont ensuite eu quelques minutes pour jouer, se familiariser avec le jeu initial, et commencer à émettre quelques critiques, ou propositions d'amélioration.

– 3. L'heure qui suit a été consacrée à la familiarisation avec le GDK. Les volontaires devaient apporter des améliorations simples, avec l'aide du professeur dans la classe. À la fin de la séance, chaque groupe a été partagé en deux équipes de 4 pour la réalisation des projets.

– 4. Les étudiants ont ensuite eu une semaine pour faire évoluer le jeu collaborativement et à distance selon les consignes. La durée total de travail conseillée pendant la semaine était de 1h30.

– 5. Une fois le travail terminé, les étudiants de chaque équipe ont reçu le jeu fait par l'autre équipe du même groupe pour pouvoir y jouer.

Afin d'évaluer l'impact de notre proposition, les deux groupes n'ont pas travaillé dans les mêmes conditions. D'un côté, le groupe 2 avait à sa disposition plusieurs éléments de la plate-forme conçue (le jeu, le GDK pour le modifier, un système de conversations contextuelles, et un système de partage de projet), c'était le groupe test. De l'autre côté, nous n'avons fourni que le jeu et le GDK au

groupe 1, qui était donc le groupe témoin. Ses membres ont donc dû trouver des moyens pour communiquer et échanger leurs réalisations.

Implémentation de notre *learning game* 2.0. Pour supporter l'activité proposée, nous avons implémenté l'application décrite dans la partie 3.3. Pour construire rapidement un prototype nous avons décidé de combiner plusieurs logiciels existants, répondant chacun à une partie des exigences. L'éditeur Game Develop a été choisi comme base de notre application. Son système de règles (conditions à gauche, actions à droite) correspond parfaitement au modèle décrit sur la figure 1. Un large panel de conditions et d'actions incluses le rend assez expressif pour créer toutes sortes de jeux en 2D. Par ailleurs, l'affichage des règles sous forme de texte en français le rend compréhensible par les personnes sans compétences en programmation, comme le montre la figure 5.

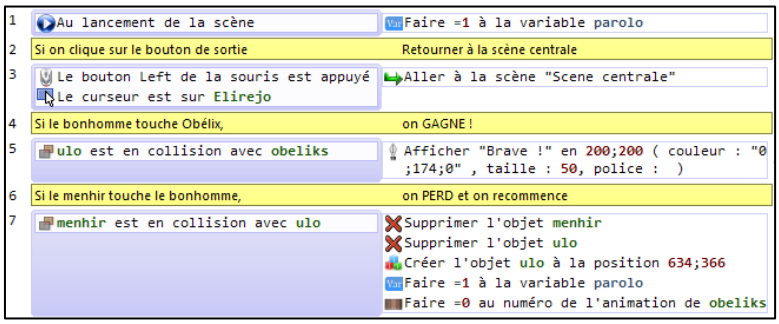

Figure 5 • Un extrait de l'éditeur de règles de Game Develop (les bandes teintées sont des commentaires)

Pour les discussions, nous avons créé un mini forum consultable dans un navigateur, à l'aide de HTML, PHP et MySQL. En parallèle, le projet d'une équipe était géré avec un système de gestion de versions (SVN) qui permettait à plusieurs utilisateurs de travailler à distance sur un projet commun, tout en maintenant à jour ses différents fichiers. La procédure à suivre par les participants était donc la suivante à chaque utilisation :

- 1. UPDATE (mise à jour du projet local grâce au projet collectif).
- 2. Recherches, réalisation de modifications, participation aux discussions, etc.
- 3. COMMIT (mise à jour du projet collectif grâce au projet local)

Dans un souci d'unification des outils, un programme servant de référence a été créé en C. Son rôle était de démarrer Game Develop (sur le projet de l'équipe) et un navigateur (sur le forum de l'équipe) en un seul clic.

4.2. Analyse des résultats

Nous analysons ici les travaux des volontaires et les résultats de l'enquête de manière globale. Un exemple de réalisation sur la figure 6 :

Figure 6 • L'état initial de la scène (gauche), et un exemple de réalisation des volontaires (droite)

Sur la figure 6, nous pouvons observer deux types de modifications effectuées par les élèves :

– Ils ont changé le contenu des messages dans les bulles des personnages, ce qui les a amené à faire des recherches sur l'Espéranto, et apprendre par eux même comment faire une conversation dans cette langue.

– Ils ont ajouté des animations liées à la conversation, pour développer le côté amusant du jeu.

Accessibilité. Pour savoir si le GDK est accessible à tous, nous avons demandé aux volontaires s'ils avaient déjà programmé (cf. Tableau 1). Nous avons ensuite observé leurs créations, et avons fait la distinction entre deux niveaux pour le travail technique réalisé, et trois niveaux pour le travail linguistique réalisé.

Groupe	1 (groupe témoin)								1 (groupe test)							
Nom de l'équipe	A				B				C				D			
A déjà programmé auparavant		P	P					P	P				P		P	P
Modifications simples	T	T	T	T	T	T			T	T	T	T				T
Modifications avancées	T				T				T	T	T				T	
Nouveau vocabulaire	E	E	E	E	E	E	E		E	E	E	E				E
Notions de cours	E				E		E		E	E	E	E				E
Nouvelles notions					E				E	E		E				

Tableau 1 • Participation au projet en fonction des compétences en programmation. (P = compétences en programmation, T = travail technique, E = travail sur l'Espéranto)

D'après les résultats, le fait de savoir programmer n'a pas été un facteur déterminant le niveau d'implication dans le projet. Aussi, un membre de l'équipe D n'a pas répondu aux questionnaires. En conséquence, l'ensemble des statistiques qui suivent s'exprimera sur un total de 15 volontaires au lieu de 16.

Motivation. Le plaisir est reconnu comme un facteur d'apprentissage important. C'est pourquoi notre proposition est basée sur un jeu. Nous avons donc demandé aux volontaires s'ils s'étaient amusés (figure 7).

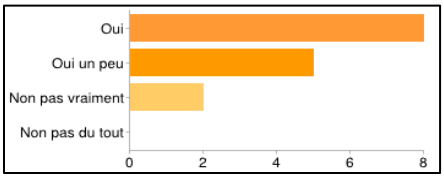

Figure 7 • Réponses à la question « Est-ce que l'activité vous a plu ? »

Nous avons également obtenu des réponses encourageantes à cette question : « *L'objectif de l'activité était d'apprendre l'Espéranto de manière ludique. Pensez-vous qu'il a été atteint ?* »

– « *Oui, car j'ai considéré que je jouais donc du coup je me suis amusée à essayer de faire avancer le jeu. Je me suis rendue compte une fois l'expérimentation terminée que j'avais très bien retenu ce que j'avais appris de manière ludique.* »

– « *Oui. Je ne suis pas du tout littéraire, et les langues ne sont pas ma passion, or ici j'ai pris plaisir à cette expérience.* »

– « *Oui, car il nous a permis d'intégrer des notions, sans pour autant avoir l'impression d'avoir beaucoup travaillé dessus. Mais on est encore loin d'avoir un niveau convenable.* »

La méthode semble perfectible, mais permet effectivement d'apprendre avec plaisir. Nous nous attendions aussi à ce que le fait de produire quelque chose d'utile (un jeu vidéo) soit une source de motivation. Les volontaire nous ont confié : « *On a l'impression que ça sert à plus que seulement nous apprendre à nous !* ». D'autres parlent d'un « *sentiment d'utilité* », et trouvent que « *c'est valorisant* ». L'implémentation concrète des savoirs a donc été un élément motivant.

Apprentissage. L'objectif le plus important de l'activité de *modding* était d'amener à un apprentissage collaboratif. Nous avons alors posé aux personnes ayant participé à l'expérimentation des questions sur l'apprentissage. La figure 8 montre en effet que cette activité peut être une méthode d'enseignement intéressante.

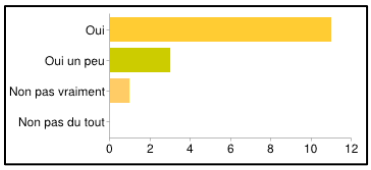

Figure 8 • Réponses à la question « Est ce que l'activité vous a fait apprendre des choses sur Espéranto ? »

Nous espérions aussi que les apprenants enrichissent leurs savoirs mutuellement, de part le travail en groupe. Cependant, d'après la figure 9, les forums n'ont pas été un moyen d'échange des savoirs aussi utile que nous l'avions espéré. Ils ont été bien utilisés pour présenter les modifications faites, mais pas pour en débattre.

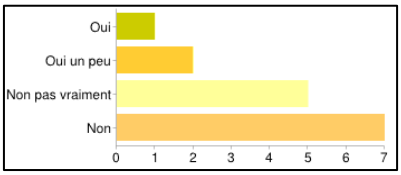

Figure 9 • Réponses à la question « Est-ce que vous pensez avoir appris des notions sur l'Espéranto en discutant avec les autres membres de l'équipe ? »

Quand un volontaire se posait une question sur l'Espéranto, il pouvait soit chercher la réponse sur internet, soit aller poser sa question dans le fil de discussion associé. Sachant que seulement 3 autres personnes consultaient le forum, il est évident que la première option était la plus rapide. Si les discussions entre participants ne peuvent pas naître de questions, elles peuvent alors naître de débats. Cependant l'expérience était trop courte (1h30) pour faire apparaître des désaccords qui causeraient un débat. Ainsi, le potentiel de l'application prendrait tout son sens dans le cadre d'une activité à plus long terme et impliquant davantage d'étudiants.

Réutilisation des jeux. Les volontaires ont été enthousiastes à l'idée de recevoir les jeux des autres équipes (au moins 5 disent l'avoir attendu impatiemment). Ils nous ont confié que « *c'était assez amusant* », ou encore « *très intéressant, parce que ça permet de voir les idées des autres, et de tester d'autres exercices sur l'Espéranto* ». Finalement, « *c'était une bonne idée, et l'apprentissage continue quand on essaie le jeu des autres* ». Par ailleurs, 10 des volontaires y ont trouvé des notions qu'ils avaient intégrées eux-mêmes dans leur jeu. Ainsi, 5 d'entre eux ont trouvé les réponses à des questions qu'ils s'étaient eux-mêmes

posées en *moddant* de leur côté. Enfin, tous ont trouvé cette dernière étape amusante.

Pendant le développement, les *moddeurs* étaient aussi amenés à tester les modifications créées par les membres de leur équipe. Globalement, la méthode des équipes A et B (envoi par mail du projet) a posé un problème en termes d'accessibilité au travail des autres, tandis que les équipes C et D disposaient en permanence du travail des autres. À la fin, tous les membres de l'équipe C disent avoir testé les modifications apportées par leurs collègues avant d'envoyer le jeu au professeur. En revanche, seulement la moitié des membres du groupe 1 l'avait fait. Cette différence explique les résultats mitigés de la figure 10, et démontre l'utilité du travail sur un projet partagé avec le SVN.

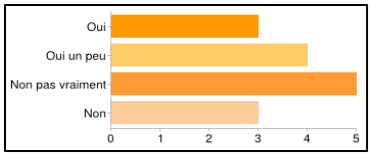

Figure 10 • Réponses à la question « *Est ce que vous pensez avoir appris des notions sur l'Espéranto en jouant aux modifications faites par les autres de votre équipe ?* »

5. Conclusion

Dans cet article, nous avons établi un état de l'art des situations d'apprentissage basées sur le *modding*. Nous avons ensuite proposé un modèle d'activité qui est une extension des *learning games* vers des *learning games* 2.0 collaboratifs. Cette activité basée sur le jeu, s'inscrit dans une approche constructiviste plaçant l'apprenant comme acteur de l'apprentissage. Nous avons défendu l'idée que cette activité peut être accessible à des publics divers si des outils appropriés sont fournis, qu'elle peut s'utiliser pour enseigner tous types de savoirs, et qu'elle est réellement intéressante si elle est pratiquée collaborativement, et supportée par une instrumentation informatique adéquate. Nous avons aussi présenté plusieurs exemples concrets pour illustrer notre proposition.

Nous avons ensuite proposé un modèle et des outils adaptés pour supporter cette activité. Nous avons finalement testé notre modèle d'activité avec un groupe de 16 volontaires. Cette étude a présenté des résultats très encourageants, car l'activité semble être un moyen d'apprendre motivant et accessible à tous. Elle a aussi mis évidence les obstacles techniques qu'il reste encore à dépasser, en particulier concernant les outils de discussion. Aussi, une évaluation comparative avec un groupe témoin a montré que le modèle de collaboration et de discussion proposé tend à favoriser l'activité d'apprentissage par *modding*.

L'expérimentation organisée nous a permis de faire les premiers pas sur le sujet encore exploratoire de la modification des jeux éducatifs. La structure informatique et l'interface sont des premières propositions qui devront être améliorées et aussi adaptées aux situations d'apprentissage (type de public, durée de l'activité, niveau d'exigence de l'apprentissage, etc.). De plus, des expérimentations à plus grande échelle pourront apporter des résultats plus quantitatifs. Les interactions entre participants n'ont pas été aussi riches que nous l'avions espéré. La faible taille des groupes et la durée restreinte de l'expérience sont certainement des éléments explicatifs, mais des pistes restent à explorer pour mieux favoriser la collaboration à distance.

Si un prototype a pu être réalisé, plusieurs défis techniques restent à relever pour que les outils soient à la hauteur du modèle proposé, en particulier l'intégration de toutes les facettes de l'activité au sein d'une plate-forme unique. Aussi, nous avons proposé de nombreux outils pour les apprenants, mais très peu pour les enseignants. Pourtant nous leur demandons beaucoup, (conception de la carte des connaissances, et mise en relation de cette carte avec les éléments du jeu) sans encore proposer d'outils adéquats. Enfin, la question de l'évaluation devra aussi être abordée, afin que le processus de correction soit intégré à l'activité de *modding* éducatif.

D'une manière plus globale, nous espérons que ces travaux ouvriront la porte à de nouveaux types d'activités pédagogiques, donnant aux apprenants de puissants outils de représentation et d'expression. Ils pourront ainsi devenir maîtres de leur apprentissage, progressant chacun à leur niveau et à leur rythme, tout en travaillant ensemble.

BIBLIOGRAPHIE

(Ang *et al.*, 2005)

Ang, C. S., Zaphiris, P., Wilson, S. (2005) Wiki-supported Collaborative Narrative Construction in Game Communities, *The ECSCW'05 workshop on "Computer Games CSCW"*, Paris, France.

(Backlund *et al.*, 2011)

Backlund P., Alklind Taylor A._S., Carlén U., Engström H., Johannesson M., Lebram M. Toftedahl M. (2011) Tactical Incident Commander - an Online Training Game for Incident Commander Training. *Proceedings of the 5th European Conference on Games Based Learning*, Athenes, Greece, p. 9-17.

(Becker, 2011)

Becker, K. (2011) The Magic Bullet: A Tool for Assessing and Evaluating Learning Potential in Games, *International Journal of Game-Based Learning*, Vol 1, No. 1, p. 19–31.

(Cignoni, 2001)

Cignoni, G. (2001) Reporting about the Mod software process. *Software Process Technology*, p. 242–245.

(Djaouti *et al.*, 2010)

Djaouti, D., Alvarez, J., Jessel, J. P. (2010) Can Gaming 2.0 help design Serious Games?: a comparative study. *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games*, p. 11–18.

(Djaouti, 2011)

Djaouti, D. (2011) *Serious Game Design - Considérations théoriques et techniques sur la création de jeux vidéo à vocation utilitaire*, Thèse en Informatique, Université de Toulouse III - Paul sabatier.

(El-Nasr et Smith, 2006)

El-Nasr, M. S., Smith, B. K. (2006) Learning through game modding. *Computers in Entertainment (CIE)*, Vol 4, No. 1, Article 3B.

(George, 2004)

George S. (2004) Contextualizing discussions in distance learning systems, *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies (ICALT 2004)*, Joensuu, Finland, p. 226–230.

(Lave et Wenger, 1991)

Lave J., Wenger E. (1991) Situated learning: Legitimate peripheral participation, Cambridge University Press, New York, 138 p.

(Loh et Byun, 2009)

Loh, C. S., Byun, J. H. (2009) Modding Neverwinter Nights into serious games, *Digital Simulations for Improving Education: Learning Through Artificial Teaching Environments*, p. 408–426.

(McAtamney et al., 2005)

McAtamney, H., O'Shea, B., Mtenzi, F. (2005) Using the Crytek game engine in the Dublin Institute of Technology, *Proceedings of the 7th International Conference on Computer Games*, Angoulême, France, 5p.

(Monterrat et al., 2012)

Monterrat, B., Lavoué, E. et George, S. (2012) Learning Game 2.0: Support for Game Modding as a Learning Activity. *Proceedings of the 6th European Conference on Games Based Learning*, Cork, Ireland, p. 340-347.

(Moshirnia, 2007)

Moshirnia, A. (2007) The educational potential of modified video games. *Issues in informing science and information technology*, Vol 4, p. 511–521.

(Oblinger, 2006)

Oblinger, G. (2006) Games and Learning, *Educase quarterly*, Vol 29, No. 3, p. 1–7.

(Postigo, 2008)

Postigo, H. (2008) Video Game Appropriation through Modifications: Attitudes Concerning Intellectual Property among Modders and Fans. *Convergence: The International Journal of Research into New Media Technologies*, Vol 14, No. 1, p. 59-74.

(Scacchi, 2010)

Scacchi, W. (2010) Computer game mods, modders, modding, and the mod scene. *First Monday*, vol 15, No. 5.

(Scacchi, 2011)

Scacchi, W. (2011) Modding as a Basis for Developing Game Systems. *Proceedings of the 1st international workshop on Games and software engineering*, Waikiki, Honolulu, HI, USA, p. 5–8.

(Sotamaa, 2005)

Sotamaa, O. (2005) Critical Perspectives On Computer Game Mod Competitions. *Proceedings of DiGRA 2005 Conference: Changing Views–Worlds in Play*, Vol 16.

(Sotamaa, 2008)

Sotamaa, O. (2008) When The Game is Not Enough: Motivations and Practices among Computer Game Modding Culture. *Games and Culture*, Vol 5, No. 3, p.239–255.

(Tavares et Roque, 2007)

Tavares, J. P., et Roque, L. (2007) Games 2.0: Participatory Game Creation. *Proceedings of the 6th Symposium on Computer Games and Digital Entertainment*. São Leopoldo, Brazil.

(Volk, 2007)

Volk, D. (2007) Game development 2.0. *Proceedings of the 2007 conference on Future Play*, Toronto, ON, Canada, p. 225–228.

(Volk, 2008)

Volk, D. (2008) Co-creative game development in a participatory Metaverse. *Proceedings of the Tenth Anniversary Conference on Participatory Design 2008*, Bloomington, IN, USA, p. 262–265.