HAL
open science

Vers un jeu sérieux pour enseigner la programmation

Mathieu Muratet, Patrice Torguet, Jean Pierre Jessel, Fabienne Vallet

To cite this version:

Mathieu Muratet, Patrice Torguet, Jean Pierre Jessel, Fabienne Vallet. Vers un jeu sérieux pour enseigner la programmation. AFRV 2008, Oct 2008, Bordeaux, France. hal-01359779

HAL Id: hal-01359779

https://hal.science/hal-01359779v1

Submitted on 4 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Vers un jeu sérieux pour enseigner la programmation

Muratet Mathieu* Equipe VORTEX, IRIT
Torguet Patrice† Equipe VORTEX, IRIT
Jessel Jean-Pierre‡ Equipe VORTEX, IRIT
Viallet Fabienne§ Equipe DiDiST, CREFI-T

RÉSUMÉ

Les jeux vidéo font partie intégrante de la culture au même titre que la télévision, les films et les livres. Nous pensons que ces applications de réalité virtuelle, grandement utilisées, peuvent servir à la formation des étudiants. Ce type d'outil est appelé : jeux sérieux. Les jeux sérieux sont présents, aujourd'hui, dans de nombreux secteurs d'activité comme l'éducation, la santé, la défense, l'industrie, la sécurité civile et les sciences. Ce document présente une étude pour déterminer le type de jeu le plus adapté à l'enseignement de la programmation. Les jeux de stratégie temps réel semblent être particulièrement intéressants dans ce cadre.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; K.3.1 [Computers and Education]: Computer Uses in Education—Collaborative Learning

1 INTRODUCTION

Dans les années 80, la révolution informatique a permis l'émergence de la réalité virtuelle dans laquelle des individus réalisent une expérience immersive et interactive dans un univers de synthèse. Cette discipline a connu son origine dans le domaine militaire en permettant de préparer des individus à des situations critiques. Dans le domaine civil, elle s'est développée avec les simulateurs de conduite. Les premiers systèmes permettaient d'immerger une seule personne à la fois dans un même monde virtuel. Par la suite, les innovations technologiques ont autorisé l'interaction de plusieurs utilisateurs distants. La réalité virtuelle distribuée est donc apparue avec de nouvelles perspectives et défis à surmonter. Aujourd'hui les jeux vidéo en réseau constituent des applications singulières de la réalité virtuelle distribuée [15].

Le jeu vidéo est devenu un phénomène de société et occupe une place prépondérante sur le marché mondial. D'après les chiffres de l'ESA (*Entertainment Software Association*) [1], en 2007 le marché de l'informatique et du jeu vidéo aux Etats-Unis a atteint 9,5 milliards de dollars. Ce marché égale celui de l'industrie du cinéma[2] (9,6 milliards de dollars en 2007 aux Etats-Unis). Ce moyen de divertissement fait partie intégrante de la culture au même titre que la télévision, les films et les livres. Les technologies des jeux vidéo, appliquées à des secteurs non divertissants, résultent d'un nouveau champ de recherche en informatique : le *serious game* ou jeu sérieux. Ces nouveaux jeux permettent la simulation, l'immersion et l'interaction dans des mondes virtuels en vue de former les joueurs au domaine traité par le jeu. Le jeu vidéo est alors utilisé comme vecteur pour amener le joueur à résoudre des problèmes. Dans ce contexte d'apprentissage interactif, l'apprenant devient acteur de sa propre formation.

Dans la suite de ce document nous décrivons l'étude qui nous a permis de définir un jeu sérieux pour l'enseignement de la programmation. Dans une première partie, nous présentons les difficultés rencontrées par les étudiants lors de l'apprentissage de la programmation ainsi qu'un court état de l'art sur les outils innovants permettant cet apprentissage. Dans une deuxième partie, nous décrivons les choix effectués pour notre jeu sérieux à l'aide d'une enquête distribuée aux étudiants de notre université. Dans une troisième partie, nous détaillons la conception et l'implémentation de notre jeu sérieux et nous montrons comment il est possible d'utiliser différents langages de programmation. Enfin, nous présentons une ingénierie pédagogique utilisant cet outil pour des étudiants novices en programmation.

2 ENSEIGNEMENT DE LA PROGRAMMATION ET OUTILS ASSOCIÉS

2.1 Problèmes liés à l'enseignement de la programmation

L'enseignement de l'informatique a pour fondements l'apprentissage de l'algorithmique et des langages de programmation. Cet enseignement pose, en particulier aux étudiants novices, de réelles difficultés qui les conduisent à abandonner leur formation ou à échouer massivement. Les difficultés rencontrées sont de plusieurs ordres : (1) obstacles épistémologiques, comme la maîtrise de la construction de la boucle [4, 14] ; (2) obstacle environnemental car le contexte d'étude est désuet par rapport à celui dans lequel ils évoluent quotidiennement ; (3) obstacle pédagogique, la programmation nécessite une pratique importante pour pouvoir être maîtrisée, ce qu'ils ne font pas par manque de motivation. Des solutions pédagogiques, comme la création de langages de programmation spécifiques destinés à minimiser les difficultés, sont proposées par les enseignants mais ne suffisent pas à enrayer le taux d'échec.

2.2 Etat de l'art des outils existants

Pour résoudre ces problèmes, différents types de logiciels ont été développés. Certains d'entre eux utilisent des langages graphiques à base de blocs. Cette métaphore de programmation permet à l'étudiant de se détacher de la syntaxe afin de se concentrer sur la résolution de problème. StarLogo The Next Generation [9], Scratch [10], Alice2 [8] et Cleogo [2] utilisent cette approche.

Une autre approche consiste à utiliser la compétition pour motiver les étudiants. C'est le cas du projet Robocode[3] et de l'évènement international RoboCup[4].

La dernière solution utilise le jeu vidéo pour accrocher le joueur et l'amener à la programmation. Deux approches ont été expérimentées :

- Les étudiants développent leur propre jeu. [1, 5] sont deux exemples de cette approche.
- Les étudiants peuvent également apprendre lorsqu'ils jouent à un jeu. Le projet WISE (*Wireless Intelligent Simulation Environment*) [3] est un environnement de jeu interactif qui com-

*e-mail:Mathieu.Muratet@irit.fr
†e-mail:Patrice.Torguet@irit.fr
‡e-mail:Jean-Pierre.Jessel@irit.fr
§e-mail:Fabienne.viallet@iut-tlse3.fr

[1] http ://www.theesa.com/facts/pdfs/ESA_EF_2008.pdf accédé le 26 Août 2008

[2] http ://www.the-numbers.com/market/2007.php accédé le 26 Août 2008

[3] http ://robocode.sourceforge.net/ accédé le 17 Avril 2007

[4] http ://www.robocup.org/ accédé le 09 Avril 2007

bine jeu réel et virtuel. Colobot[5] est le seul exemple, que nous connaissons, de jeu vidéo complet qui combine interactivité, histoire et programmation. Dans ce jeu, le joueur doit coloniser des planètes en utilisant et programmant des robots.

A l'exception de Robocode, WISE et Colobot, nous ne pouvons considérer les outils précédents comme des jeux vidéo. Cependant, ils introduisent des approches intéressantes et transférables sur un jeu sérieux tel que les langages graphiques à base de bloc ou les environnements collaboratifs.

Robocode, WISE et Colobot peuvent être considérés comme des jeux sérieux, mais ne sont pas suffisamment complets. Robocode manque d'interactivité : le joueur est inactif durant la simulation, il reste spectateur de sa propre intelligence artificielle. WISE demande de nombreuses ressources (espace, robots...). Sa mise en œuvre est difficile avec de nombreuses contraintes lors de l'expérimentation. Enfin, Colobot manque d'un mode multi joueur. Or, nous pensons que le travail compétitif et collaboratif introduit par l'aspect multi joueur peut être très intéressant pour l'étudiant [7].

Notre travail s'appuie sur ces outils. Nous proposons d'enseigner la programmation au travers d'un jeu sérieux : un jeu multi joueur où la programmation devient un avantage dans la partie. Ainsi nous proposons aux étudiants un environnement d'apprentissage familier, amusant et collaboratif qui devrait susciter leur motivation et leur permettre d'améliorer leurs compétences en la matière de programmation.

2.3 Contraintes

Le point critique d'un jeu sérieux est la relation existante entre le jeu et son contenu pédagogique. Différentes expériences ont montré que les jeux sérieux atteignent leurs objectifs lorsqu'ils ont une forte composante « jeu » clairement mise en avant. C'est l'approche qui a notamment été utilisée par *America's Army* [15], le premier jeu sérieux à avoir réellement rencontré un succès auprès du public.

De plus, pour être efficace, pertinent et motivant en terme d'apprentissage, un jeu sérieux doit satisfaire plusieurs caractéristiques [6] : proposer plusieurs degrés de difficultés, s'adapter au niveau du joueur, proposer des objectifs clairs et intéressants, permettre l'interaction entre les joueurs et faire partager les connaissances.

Ainsi, le jeu sérieux que nous souhaitons concevoir doit respecter un ensemble de contraintes : le type de jeu utilisé doit être connu et apprécié des étudiants, le jeu doit être compatible avec la pratique de la programmation, les programmes développés par les étudiants doivent bonifier le jeu, le jeu doit être multi joueur et permettre un apprentissage collaboratif, le jeu sérieux doit être compatible avec différents langages de programmation, y compris des langages spécifiques développés pour les novices. Dans la suite de ce document, nous expliquerons la conception et le développement de notre jeu sérieux pour respecter ces contraintes.

3 Quel type de jeu choisir ?

La première étape de notre travail a consisté à définir le type de jeu le plus joué par nos étudiants. Nous leur avons soumis un questionnaire : 181 étudiants ont été interrogés (154 garçons et 27 filles) dans trois formations différentes (deux en informatique et une en génie civil). La moyenne d'âge est de 21 ans (voir figure 1 pour la répartition).

3.1 Les étudiants et les jeux vidéo

La première analyse des résultats montre une cohérence avec les chiffres de l'ESA à propos de l'intérêt des jeunes pour les jeux vidéo. La figure 2 montre le pourcentage d'étudiants joueurs, garçons et filles confondus : 91% des garçons et 52% des filles sont concernés par les jeux vidéo.

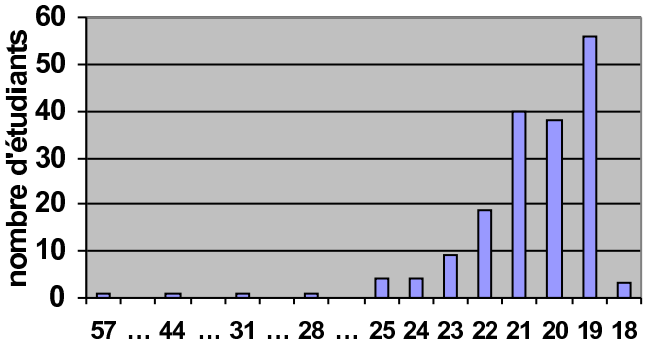

FIG. 1: Age des étudiants.

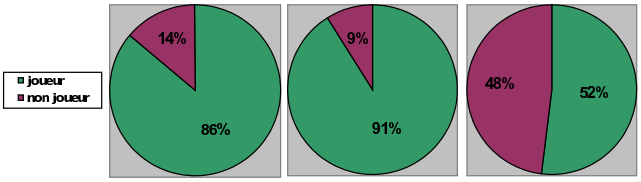

FIG. 2: Pourcentage de joueurs pour tous les participants (à gauche), les garçons (au centre) et les filles (à droite).

La seconde analyse identifie le type de jeu et le mode de jeu le plus joué par nos étudiants. Ceci nous aide à cerner le jeu approprié. Le jeu doit être attractif pour nos étudiants et adaptable à l'enseignement de la programmation.

Les jeux vidéo peuvent être classés selon différents critères. Dans cette enquête, nous avons choisi de proposer la classification habituellement utilisée par la presse du jeu vidéo. La figure 3 montre le pourcentage de joueurs pour chaque type de jeu. Le type de jeu le plus joué est le jeu de stratégie. Nous notons que ce type de jeu est également apprécié des filles (57% des filles joueuses jouent aux jeux de stratégie). Par conséquent, nous choisissons d'utiliser un jeu de stratégie comme support à notre jeu sérieux.

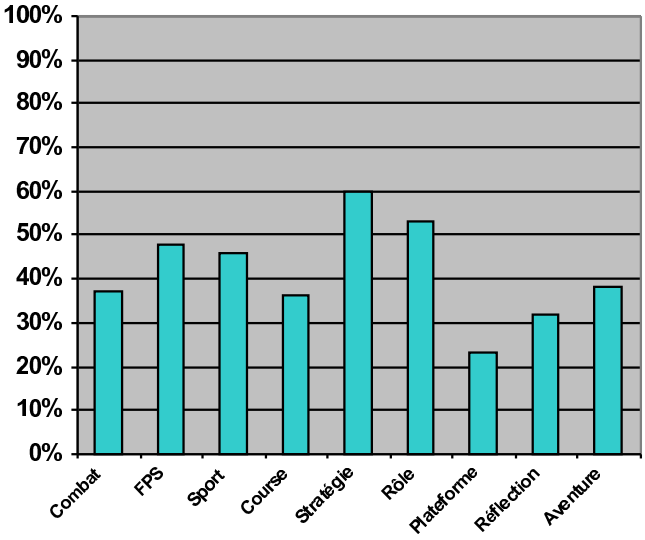

FIG. 3: Pourcentage de joueurs pour chaque type de jeu.

[5] http ://www.ceebot.com/colobot/index-e.php accédé le 21 Septembre 2007

3.2 Compatibilité entre le jeu et l'enseignement

Un jeu sérieux pour l'apprentissage de la programmation doit être compatible avec l'activité de programmation du niveau visé. Les étudiants doivent être capables d'écrire des programmes du plus simple pour les novices, au plus complexe pour des enseignements de haut niveau comme l'intelligence artificielle.

Les jeux de stratégie requièrent des compétences en planification, anticipation et réactivité. De plus, les jeux de stratégie sont intéressants pour la programmation à plusieurs niveaux :

- Pour la macro gestion : il est possible de construire des programmes qui supervisent le développement d'une armée/population. Un jeu de stratégie classique est basé sur trois phases principales : la construction, l'exploration et le combat [13]. Nous pensons que des programmes peuvent être créés pour chacune de ces phases.
- Pour la micro gestion : les programmes seraient utilisés pour gérer chaque entité du jeu. Dans un jeu de stratégie, les joueurs doivent contrôler plusieurs centaines d'unités, l'utilisation de programmes permettrait de réduire la charge cognitive du joueur.
- Pour la planification : les programmes peuvent anticiper les actions de l'adversaire. Le joueur évolue dans un environnement virtuel avec des informations incomplètes. Cet aspect important du jeu augmente la complexité des programmes liés à ce problème.

Dans un jeu de stratégie temps réel (STR), le joueur donne des ordres à ces unités pour réaliser des actions (se déplacer, construire un bâtiment...). Actuellement ces ordres sont donnés en cliquant sur une carte à l'aide de la souris. Un STR où ce type d'opération pourrait être donné grâce à un programme peut être la solution à notre jeu sérieux. Ces programmes assisterons le joueur/étudiant et augmentera sa probabilité de remporter la partie, si ses programmes sont efficaces, pertinents et adaptés au jeu. De plus, lorsqu'ils testeront leurs programmes, les étudiants utiliseront le même environnement (le jeu). Notre jeu sérieux est donc un STR où les ordres peuvent être donnés par des programmes.

3.3 Apprentissage collaboratif : jeu multi joueur

Une des caractéristiques les plus importantes de notre jeu sérieux est son mode multi joueur qui permet de développer les aspects concurrentiels et collaboratifs entre étudiants. Le choix d'un STR comme support à notre jeu sérieux permet la mise en œuvre du mode multi joueur qui est très apprécié par nos étudiants (voir figure 4) et permet d'envisager une variété d'interactions : alliances, coopération et compétition.

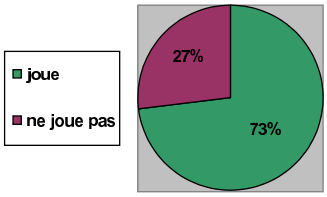

FIG. 4: Pourcentage de joueurs appréciant le mode multi joueur.

4 MISE EN ŒUVRE

4.1 Le Cœur du jeu de stratégie temps réel multi joueur

Nous avons décidé d'utiliser le STR pour plusieurs raisons : (1) c'est le type de jeu le plus populaire parmi nos étudiants, (2) il est compatible avec l'enseignement de la programmation et (3) il permet un apprentissage collaboratif. Cependant, notre objectif n'est pas de développer un nouveau STR. Nous avons donc recherché un jeu existant utilisable comme point de départ. Ce jeu doit être amusant et doit supporter nos modifications afin de satisfaire les contraintes que nous nous sommes fixées. Heureusement, quelques projets aux codes sources ouverts sont en cours de développement. C'est le cas des projets Spring[6] et Open Real-Time Strategy (ORTS) qui sont tous deux des jeux de stratégie temps réel 3D, multi joueur.

Spring est plus abouti qu'ORTS. Une communauté de joueurs y joue tous les jours sur Internet. Cette communauté aide à découvrir les bugs qui sont par la suite corrigés par le groupe de développeurs. Ce processus n'est pas présent pour ORTS qui en est à un stade plus expérimental.

Spring est un moteur de jeu développé pour réutiliser les données issues d'un STR commercial appelé Total Annihilation. Toutefois, Spring supporte un système de « mods ». Traditionnellement, les « mods » constituent des ajouts destinés à modifier la manière de fonctionner d'un jeu donné. Cependant, ce terme utilisé avec Spring fait référence à un jeu complet fonctionnant sur le moteur du jeu[7]. De nombreux « mods » sont actuellement disponibles pour Spring. Nous choisissons d'utiliser « Kernel Panic »[8] (voir figure 5). Ce jeu, totalement libre, définit un STR simplifié :

- pas de gestion de ressources excepté le temps et l'espace ;
- toutes les unités sont gratuites à créer ;
- petit arbre technologique avec moins de dix unités ;
- affichage graphique simple et vectoriel, qui correspond à l'univers du jeu.

Ces caractéristiques mettent l'accent sur la stratégie et la tactique pour un jeu orienté action, accessible et convivial.

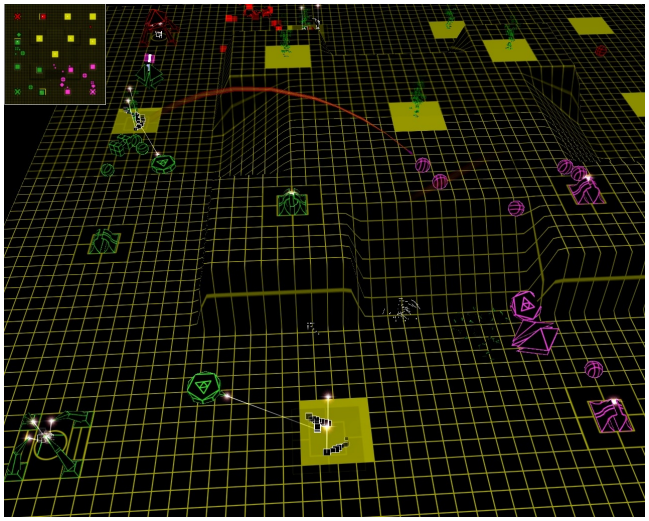

FIG. 5: Kernel Panic.

4.2 Polyvalence à différents langages de programmation

Comme nous l'avons vu précédemment, différents langages peuvent être choisis pour l'apprentissage de la programmation. Pour des raisons pédagogiques, certains enseignants ont même construit leur propre langage de programmation. Notre jeu sérieux doit donc pouvoir supporter plusieurs langages de programmation pour être utilisable dans différentes formations.

Dans nos précédents travaux [11] nous avons développé un système basé sur une bibliothèque dynamique. Dans cette version, le moteur du jeu est protégé contre les erreurs de programmation

[6] http ://spring.clan-sy.com/ accédé le 2 Février 2007
[7] http ://spring.clan-sy.com/wiki/Mods accédé le 26 Août 2008
[8] http ://spring.clan-sy.com/wiki/Kernel_Panic accédé le 26 Août 2008

du joueur et permet une prise en compte dynamique des modifications du programme. De plus, la complexité du jeu est cachée afin de faciliter la programmation.

Cependant, l'utilisation d'une bibliothèque dynamique s'est avérée être difficilement évolutive en particulier pour les langages interprétés. En effet, la bibliothèque dynamique permet de résoudre le problème dans un seul processus : le moteur du jeu. Ce processus gère l'exécution du programme de l'étudiant. Or, les langages interprétés requièrent un interpréteur qui s'exécute dans un processus indépendant.

Dans la nouvelle version, le programme de l'étudiant compilé ou interprété est indépendant du moteur de jeu et communique avec le jeu via une mémoire partagée (voir figure 6). Pour développer leur propre programme, les étudiants utilisent une librairie appelée IMJ : Interface du Moteur de Jeu. L'IMJ cache la complexité de synchronisation avec la mémoire partagée. A la demande du programme de l'étudiant, les données pertinentes sont copiées dans la mémoire partagée définie comme étant l'état du jeu. Pour éviter des situations incohérentes, le programme de l'étudiant travaille sur cette copie. Le programme lit les données et écrit les commandes à travers l'IMJ. La mémoire partagée est régulièrement vérifiée par le moteur de jeu pour réaliser les actions en attente. Tous les langages capables d'utiliser une bibliothèque C peuvent utiliser l'IMJ et communiquer avec le moteur de jeu.

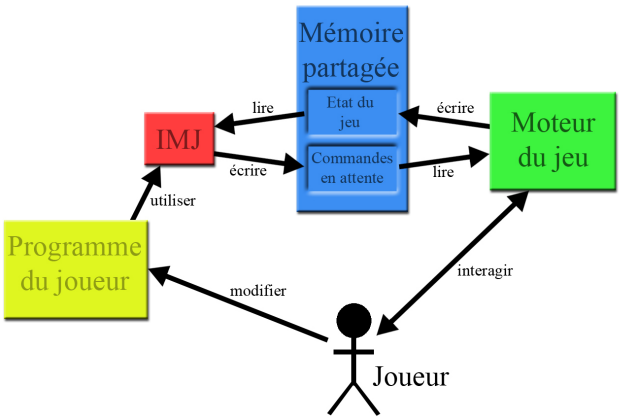

FIG. 6: Architecture.

5 EXEMPLE D'UTILISATION

Pour être efficace en terme d'apprentissage, cet outil doit être utilisé dans un cadre pédagogique approprié. En effet, cet outil n'est pas un simple jeu éducatif et les étudiants, surtout les novices, ont besoin d'être aidé par les enseignants pour écrire leurs premiers programmes et utiliser le jeu avec leurs programmes. Nous avons donc élaboré une ingénierie pédagogique destinée à des étudiants novices de première année d'IUT informatique. Le Tableau 1 présente le programme des séances prévues. Chaque séance dure deux heures et est supervisée par deux enseignants : un spécialiste du jeu vidéo et un enseignant d'informatique.

Lors de la première séance, les étudiants jouent au jeu. Une discussion sur les améliorations à apporter au jeu et sur les différentes stratégies de jeu possible est proposée. La deuxième séance est une présentation de l'interface de programmation. L'enseignant en informatique propose un algorithme à implémenter par les étudiants dans le jeu. Les obstacles à la programmation sont traités avec l'enseignant. Lors des deux semaines suivantes, les étudiants travaillent en autonomie avec l'assistance des enseignants pour développer leurs propres programmes. Dans le cas où les étudiants manqueraient d'imagination, une base de données d'algorithmes leur est proposée. L'enseignant spécialiste du jeu vidéo oriente les étudiants vers différentes stratégies de jeu afin d'enrichir les parties et répond aux problèmes liés à l'installation du jeu. Les étudiants sont autorisés à communiquer les uns avec les autres. Ils peuvent ainsi élaborer des alliances ou des stratégies coopératives, ou simplement s'entraider dans la programmation. Lorsque tous les programmes sont réalisés et opérationnels, la troisième séance a lieu : les étudiants jouent en utilisant leurs propres programmes. L'enseignant spécialiste du jeu vidéo porte son attention sur le comportement des étudiants et leur interaction avec le jeu : le rôle des programmes, l'activité des étudiants, les stratégies mises en œuvre. Cette observation est la base de la dernière séance : étudiants et enseignants analysent le jeu et tentent de trouver les raisons d'une victoire ou d'une défaite. Une discussion sur l'importance des programmes écrits par les étudiants est engagée, ainsi qu'une réflexion sur l'aspect éthique du jeu. L'objectif est que nos étudiants puissent avoir envie de continuer à utiliser ce jeu sérieux par eux-mêmes afin d'améliorer leurs compétences en programmation.

Cette ingénierie pédagogique sera testée au cours de l'année à venir dans notre université. Pour l'évaluer, nous envisageons d'utiliser plusieurs indicateurs comme l'investissement des étudiants, le nombre, la qualité et la pertinence des programmes écrits, les compétences acquises et les résultats aux examens. Nous souhaitons également utiliser le « feelgood factor » défini en [12].

La figure 7 présente un exemple de programme qui peut être écrit par un étudiant. Son objectif est de permettre aux unités de rechercher automatiquement ses adversaires. Son principe est de déterminer aléatoirement des positions où chaque unité se déplace jusqu'à ce qu'elle découvre l'adversaire. Ce programme utilise les fonctionnalités de l'IMJ, comme par exemple le sous-programme IMJ_Unité_Déplacer(u, pos) qui déplace l'unité « u » à la position « pos ». Ce programme est intéressant pour le joueur car sans lui, cette opération doit être réalisée manuellement à l'aide de la souris en sélectionnant chaque unité une par une. Ce traitement est long et fastidieux. De plus, pendant que le joueur sélectionne ses unités avec la souris, il ne peut réaliser d'autres taches, comme par exemple développer sa base.

```
#include "IMJ.h"

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main (){
        IMJ_Pos pos;
        bool trouve = false;
        srand (time (NULL)); /* Initialisation du générateur */
        while (!trouve){
                IMJ_Rafraichir();
                /* Vérfier si on a pas trouvé un ennemi */
                if (IMJ_ObtenirNbUnites(ENNEMI) > 0){
                        /* stopper toutes les unités */
                        int i;
                        for (i = 0 ; i < IMJ_ObtenirNbUnites(MOI) ; i++){
                                if (IMJ_Unite_Stop(IMJ_ObtenirUniteNum(i, MOI)) == -1){
                                        printf ("%s", IMJ_ObtenirErreur());
                                        exit (1);
                                }
                        }
                        trouve = true;
                }
                else{
                        /* parcourir toutes nos unités */
                        int i;
                        for (i = 0 ; i < IMJ_ObtenirNbUnites(MOI) ; i++){
                                /* choisir un destination aléatoire si elle est arrêtée */
                                bool deplacement;
                                if (IMJ_Unite_EstEnDeplacement(IMJ_ObtenirUniteNum(i, MOI), &deplacement) == -1){
                                        printf ("%s", IMJ_ObtenirErreur());
                                        exit (1);
                                }
                                if (!deplacement){
                                        pos.x = (rand() * (IMJ_TailleCarte().x) / (RAND_MAX+1.0));
                                        pos.y = (rand() * (IMJ_TailleCarte().y) / (RAND_MAX+1.0));
                                        if (IMJ_Unite_Deplacer(IMJ_ObtenirUniteNum(i, MOI), pos) == -1){
                                                printf ("%s", IMJ_ObtenirErreur());
                                                exit (1);
                                        }
                                }
                        }
                }
                usleep(10);
        }
        return 0;
}
```

FIG. 7: Exemple de programme pour chercher l'adversaire.

TAB. 1: Programme de l'ingénierie pédagogique.

Session 1	Session 2	Deux semaines	Session 3	Session 4
Présentation de la version initiale du STR, suivie de parties de jeu en multi joueurs	Présentation de l'IMJ et écriture d'un premier programme qui interagit avec le jeu	Les étudiants développent seuls leurs propres programmes, avec l'aide en ligne éventuelle des enseignants ou de leurs camarades (stratégie de coopération)	Parties de jeu multi joueur où chaque étudiant utilise les programmes qu'il a écrit	Remédiation : analyse du jeu, étude des programmes écrits

6 CONCLUSION

Dans ce document, nous posons les jeux sérieux comme une application pertinente de la réalité virtuelle distribuée. Ils simulent des environnements virtuels interactifs et immergent ses utilisateurs dans des mondes synthétiques. De plus, nous décrivons une étude pour déterminer le meilleur type de jeu pour enseigner la programmation. Nous choisissons d'employer un STR, car ce type de jeu est très utilisé par nos étudiants et il est compatible avec la programmation. Tous langages capables de charger une bibliothèque C peuvent servir à enseigner la programmation avec ce jeu. Nous avons vérifié que nos améliorations pouvaient être facilement intégrées à d'autres STR (comme ORTS par exemple). Ceci nous permet de choisir le meilleur jeu en fonction de nos objectifs futurs et de suivre l'évolution rapide des jeux vidéo. Cependant, cet outil n'est pas un simple jeu éducatif. Pour être efficace en termes d'apprentissage, il doit être intégré à un enseignement et à une ingénierie pédagogique.

La prochaine étape consiste à mener l'expérimentation. Nous avons commencé à travailler avec le service pédagogique de notre université et bien sûr les professeurs et étudiants des formations concernées. Les analyses expérimentales vérifieront son utilisabilité et son efficacité. D'un point de vue didactique, il sera intéressant d'analyser comment l'introduction du jeu vidéo conditionne l'activité de programmation. Une analyse comparative des tâches à réaliser dans le contexte de TP traditionnel et de TP avec l'outil permettra de déterminer ce qui est réellement enseigné.

RÉFÉRENCES

[1] W.-K. Chen and Y. C. Cheng. Teaching object-oriented programming laboratory with computer game programming. *Education, IEEE Transaction on*, 50 :197–203, 2007.

[2] A. Cockburn and A. Bryant. Cleogo : Collaborative and multi-metaphor programming for kids. In *APCHI '98 : Proceedings of the Third Asian Pacific Computer and Human Interaction*, page 189, Washington, DC, USA, 1998. IEEE Computer Society.

[3] D. J. Cook, L. B. Holder, M. Huber, and R. Yerraballi. Enhancing computer science education with a wireless intelligent simulation environment. *Journal of Computing in Higher Education*, 16(1), 2004.

[4] G. David. On novice loop boundaries and range conceptions. *Computer Science Education*, 14(3), 2004.

[5] P. Gestwicki and F.-S. Sun. Teaching design patterns through computer game development. *J. Educ. Resour. Comput.*, 8(1) :1–22, 2008.

[6] F. L. Greitzer, O. A. Kuchar, and K. Huston. Cognitive science implications for enhancing training effectiveness in a serious gaming context. *J. Educ. Resour. Comput.*, 7(3) :2, 2007.

[7] R. T. Johnson and D. W. Johnson. An overview of cooperative learning. *Creativity and Collaborative Learning*, 1994.

[8] C. Kelleher, D. Cosgrove, D. Culyba, C. Forlines, J. Pratt, and R. Pausch. Alice2 : Programming without syntax errors, 2002. UIST.

[9] E. Klopfer and S. Yoon. Developing games and simulations for today and tomorrow's tech savvy youth. *TechTrends*, 49(3) :33–41, May 2004.

[10] J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, and M. Resnick. Scratch : A sneak preview. In *C5 '04 : Proceedings of the Second International Conference on Creating, Connecting and Collaborating through Computing*, pages 104–109, Washington, DC, USA, 2004. IEEE Computer Society.

[11] M. Muratet, P. Torguet, and J.-P. Jessel. Learning programming with an rts based serious game. In *Serious Games on the Move, Cambridge, UK, 23/06/08-24/06/08*, page (electronic medium), http ://www.cambridge.org/, juin 2008. Cambridge University Press.

[12] M. M. Müller and F. Padberg. An empirical study about the feel-good factor in pair programming. In *METRICS '04 : Proceedings of the Software Metrics, 10th International Symposium*, pages 151–158. IEEE Computer Society, 2004.

[13] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu. The effect of latency on user performance in warcraft iii. In *NetGames '03 : Proceedings of the 2nd workshop on Network and system support for games*, pages 3–14, New York, NY, USA, 2003. ACM Press.

[14] E. Soloway, J. Bonar, and K. Ehrlich. Cognitive strategies and looping constructs : an empirical study. *Commun. ACM*, 26(11) :853–860, 1983.

[15] M. Zyda. From visual simulation to virtual reality to games. *Computer*, 38(9) :25–32, 2005.