L'intégration des *serious games* dans les cours d'introduction de l'algorithmique et de la programmation

Integration of serious games in teaching an introductory course on algorithmic and programming

Ibrahim Ouahbi (1,2), Fatiha Kaddari (1), Hassane Darhmaoui (2), Abdelrhani Elachqar (1)
1. Laboratoire de Didactique, d'Innovation Pédagogique et Curriculaire (LADIPEC), Faculté des sciences Dhar Almahraz, Université sidi Mohammed Ben Abdellah, Fès Maroc.
2. Center for Learning Technologies (CLT), Al Akhawayn University in Ifrane, Ifrane, Morocco

Résumé

Les élèves débutants en programmation sont confrontés à beaucoup de difficultés (multiplicité de concepts théoriques et techniques, construction de programmes et d'algorithmes…), ce qui mène à leur démotivation pour apprendre la programmation. Afin de stimuler l'intérêt des élèves, nous avons élaboré des séquences pédagogiques en algorithmique et programmation basées sur les serious games. Ainsi, une expérimentation a été menée avec un groupe expérimental de 20 élèves de niveau de la 1ère année du secondaire qualifiant où les élèves ont été initiés à des serious games : jeux Blockley, Light-Bot…, puis à la création de jeux avec Scratch un environnement visuel facilitant la programmation. Le groupe témoin, de 20 élèves aussi, a suivi une introduction à l'algorithmique et à la programmation en utilisant une méthode classique. A la fin de l'expérimentation, les deux groupes ont été amenés à répondre à un test dans le cadre des évaluations sommatives et à un questionnaire destiné à évaluer leur motivation et intérêt pour l'apprentissage de la programmation. Les résultats obtenus montrent que les élèves du groupe expérimental ont mieux acquis les notions de base de l'algorithmique et de la programmation. Nous concluons que l'innovation pédagogique via les serious games peut être un moyen pour aboutir à un apprentissage efficace des élèves, les rendre plus autonomes dans la construction de leurs savoirs et de les motiver à poursuivre leurs études en programmation.

Mots clés : *serious games*, programmation pour novices, algorithmique, environnement Scratch

Abstract

Beginning programming students face a number of learning difficulties due to a multiplicity of theoretical and technical new concepts in program construction and algorithms, which often lead to their lack of motivation to learn programming. We have developed pedagogical learning sequences based on serious games for the module algorithms and programming. We implemented our new method in an experimental class of 20 high-school students. We first introduced serious games to the students, then taught them how to create simple games in a visual environment facilitating programming (Scratch). Twenty other students (control group), followed an introduction to algorithms and programming using a conventional teaching method. Both groups had the same summative assessments as well as a survey to assess their motivation and interest in learning programming. The experimental group not only outperformed the control group but showed better motivation for programming. We conclude that serious games help achieve an effective student learning environment where students build their own knowledge and skills, and motivate them for computer sciences.

Keywords: *serious games, introductory programming, algorithmic, Scratch environment*

I. Introduction et problématique

Le statut de l'informatique en tant que "discipline scolaire" est actuellement quasiment admis par tous les systèmes éducatifs de par le monde. La tendance est d'enseigner le codage et la programmation dès le primaire, la pensée informatique et la résolution des problèmes doivent être des pierres angulaires dans les curricula de l'enseignement de l'informatique (Ouahbi et al., 2015). Cependant selon plusieurs chercheurs, les modules d'algorithmique et de la programmation figurent parmi les cours les plus redoutés et abandonnés par les élèves car les taux de décrochage sont généralement plus élevés que les autres cours (Ala-Mutka, 2012 ; Wilson et al., 2011 ; Lahtinen et al., 2005 ; Robins et al., 2003).

En effet, les concepts fondamentaux de la programmation présentent plusieurs difficultés et obstacles. On peut citer par exemple les difficultés relatives à la construction des programmes (Lahtinen et al., 2005), la manipulation des boucles (Ginat, 2004), les structures de contrôle et les algorithmes (Seppälä et al., 2006).

Certaines études stipulent que les difficultés d'apprentissage de l'algorithmique et de la programmation proviennent et/ou sont amplifiées par le manque d'innovation dans les méthodes d'enseignement, la démotivation des élèves et le manque d'interaction avec et entre les élèves (Brito et al., 2014 ; Barker et al., 2009 ; Crenshaw et al., 2008). Plus particulièrement, le désintérêt des élèves débutants est attribué au fait qu'ils sont confrontés dès le début à un champ qui regorgent de concepts théoriques et techniques (Bennedsen et al., 2008 ; Matocha et al., 1998).

Pour dépasser ces difficultés, plusieurs réponses ont été suggérées dans la littérature :

- Kelleher et al. (2005) et Muratet et al. (2009) ont proposé de faire évoluer les environnements et langages de programmation pour les novices. La plupart de ces environnements utilisent des commandes graphiques à base de blocs. L'intérêt de ces environnements est de permettre à l'élève débutant de se concentrer sur l'algorithmique sans se soucier de la syntaxe (Muratet et al., 2009).
- L'utilisation des jeux vidéo et en particulier les serious games pour apprendre a également été envisagée (Kazimoglu et al., 2012 ; Chaffin et al., 2009 ; Barnes et al., 2007). En effet, le recours aux serious games permet de motiver les élèves et susciter leur intérêt pour la programmation. Ainsi, en s'appuyant sur la passion des jeunes pour le numérique, on les incite à développer de nouvelles connaissances dans le système scolaire (Ouahbi et al., 2014).

Ces résultats et constats nous amènent à formuler la question de recherche suivante :

« Une utilisation efficace des serious games, peut-elle pallier aux difficultés rencontrées dans l'enseignement/apprentissage des notions de base de l'algorithmique et de la programmation? »

L'accent a été mis sur l'enseignement/apprentissage de l'algorithmique et de la programmation aux élèves de la 1ère année du secondaire qualifiant au Maroc. Nous visons vérifier l'hypothèse suivante :

"L'usage des serious games et la création des jeux à l'aide d'un environnement facilitant la programmation tel que Scratch, favorise un apprentissage efficace du module algorithmique et programmation".

Pour élaborer une expérimentation adéquate, nous nous sommes appuyés sur un cadre théorique explicitant la terminologie relative aux serious games et les différentes stratégies développées pour l'intégration de ces jeux en classe. L'expérimentation qui consistait à pratiquer et à créer des serious games avec l'environnement Scratch a été menée auprès de 40 élèves du niveau de la 1ère année du cycle secondaire qualifiant scientifique. Les données de cette dernière corroborent bien celles de la littérature à savoir : le serious game peut être un outil efficace d'enseignement et d'apprentissage.

II. Cadre théorique

A. Serious games

1. Origine de l'oxymore serious game

L'expression serious game apparaît comme un oxymore[1] rassemblant deux mots en apparence contradictoire qui sont le jeu et le sérieux. En effet, les définitions classiques du terme jeu opposent souvent le jeu au sérieux (Schmoll, 2013; Mutet, 2003). Cependant, plusieurs chercheurs comme Schiller (1943), Freud (1985) et Henriot (1989) estiment que ces deux termes ne sont pas si opposés qu'on ne l'on pense.

L'origine du terme serious game remonte à l'époque de la renaissance (XV^{ème} siècle) (Djaouti et al., 2011). L'expression "*serio ludere*" employée à cette époque faisait référence à l'utilisation de l'humour dans la littérature en vue de la transmission des notions sérieuses.

Dans le domaine de l'éducation et de l'enseignement, le premier à avoir eu recours au terme serious game est Clark Abt. En effet, dans son ouvrage intitulé "*Serious Games*" (Abt, 1970), ce chercheur avait souligné que les jeux sont un parfait outil pour diffuser des messages éducatifs, politiques, marketing etc. Cependant ces jeux, qualifiés de sérieux, selon Abt ne sont pas forcément liés à un support informatique. Actuellement, le terme serious game est généralement utilisé dans un contexte informatique. Cette limitation des serious games aux jeux vidéo et aux supports informatiques a été initiée par les travaux de Ben Sawyer et David Rejeski : auteurs du livre blanc "*Improving Public Policy through Game Based Learning and Simulations*" (Sawyer et al., 2002)[2]. Selon Schollmeyer (2006), le terme serious game émane des travaux de ces deux chercheurs.

En 2002, Ben Sawyer et David Rejeski ont fondé l'association "*Serious Games Initiative*" qui avait pour objectif celui de tisser des liens productifs entre l'industrie du jeu électronique et les projets impliquant l'utilisation des jeux dans les domaines de l'éducation, la formation, la santé et des politiques publiques (Susi et al., 2007). Cette même année a marqué la naissance officielle du serious game aux Etats Unis par la sortie du jeu "*America's Army*"[3]. Ce jeu a été considéré par Ben Sawyer comme étant le premier serious game à avoir massivement attiré l'intérêt d'un public assez large.

En résumé, l'année 2002 peut être considérée comme l'année de l'avènement du serious game (Alvarez, 2007; Djaouti, 2011; Zyda, 2005; Sawyer, 2007). Depuis, de nombreux serious games ont été réalisés dans des domaines variés. En conséquence plusieurs définitions et appellation du terme serious game ont été proposées (Marfisi-Schottman, 2013).

2. Définition du terme serious game

Le terme serious game est communément utilisé dans un contexte informatique, pour désigner une gamme de jeux vidéo dont la finalité première est autre que le simple divertissement *(*Michael et al., 2005*)*. Le chercheur et le "*game designer*" Zyda, un des participants au développement du jeu "*America's Army*" propose la définition suivante : "*un défi cérébral contre un ordinateur selon des règles spécifiques, qui utilise le divertissement pour servir à la formation institutionnelle ou professionnelle, l'éducation, la santé, la politique intérieure et la communication*"[4] (Zyda, 2005). Dans le même courant, Sawyer (2007) définit le serious game par : « *toute utilisation pertinente des technologies issues de l'industrie du jeu vidéo à des fins autres que le divertissement* »[5].

[1] Figure de style qui vise à rapprocher deux termes que leurs sens devraient éloigner, dans une formule en apparence contradictoire. Définition prise à partir de https://fr.wikipedia.org/wiki/Oxymore, le 2/8/2015.
[2] David Rejeski a modifié le titre de ce libre blanc en incluant le terme serious games.
[3] http://www.americasarmy.com/, consulté le 10/02/2015
[4] "*Serious game: 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*."
[5]"*Any meaningful use of computerized game/game industry resources whose chief mission is not entertainment*"

Après avoir analysé les définitions recensées dans la littérature, principalement celles de Sawyer et Zyda, Alvarez propose une définition synthétique qui implémente une caractéristique spécifique aux serious games à savoir le scénario pédagogique : le serious game est une « *[a]pplication 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 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"* » (Alvarez, 2007, page 51).

A partir des définitions déjà citées ci-dessus, il apparaît selon nous que les serious games représentent le plus souvent une gamme particulière de jeux vidéo. Ce qui différencie un serious game d'un jeu vidéo de divertissement est le fait que les serious games sont explicitement et intentionnellement conçus à des fins sérieuses. Pourtant, rien n'empêche d'utiliser les jeux vidéo de divertissement à des fins sérieuses. En effet, l'utilisation des jeux vidéo à des fins sérieuses n'est pas une innovation récente : on peut procéder à un détournement des règles premières d'un jeu vidéo classique pour en ressortir des applications utiles. Cet usage des jeux vidéo de divertissement vers une finalité utilitaire est désigné par le terme "*Serious gaming*" (Kasbi, 2012).

La notion de *serious gaming* ne se limite pas uniquement aux jeux vidéo de divertissement, on peut détourner aussi l'usage d'une application utilitaire à des fins ludiques (Michaud et al., 2008). L'application *Microsoft Word* peut être utilisée pour le divertissement, l'utilisateur peut s'amuser à taper des lettres et modifier leur mise en forme d'une façon aléatoire. Dans la même lignée, avec l'application *Paint* un débutant qui trouve des difficultés avec la manipulation de la souris peut se divertir en dessinant et coloriant des formes géométriques.

En effet, on peut dire que toute application utilitaire peut potentiellement divertir. Ainsi l'évaluation de la nature ludique ou sérieuse d'une application reste subjective. Certains chercheurs stipulent que c'est l'intention du concepteur qui différencie un serious game d'un jeu vidéo de divertissement (Adams, 2010). Pour notre part, nous pensons que cela relève également de l'attitude de l'utilisateur et de son usage de l'application. Cela nous renvoie aux travaux de Frasca, qui souligne que « *c'est l'utilisateur et non pas le concepteur qui décide comment utiliser un jeu ou un jeu vidéo. Le concepteur peut suggérer un ensemble de règles, mais le joueur a toujours la décision finale* »[6] (Frasca, 2003)

En général, l'ensemble des définitions du terme serious game ci-dessus, s'accordent sur le fait que ce dernier est caractérisé par l'association de deux aspects : sérieux (implicite) et ludique. La liaison du serious game à un support informatique nous paraît logique vu la popularité et l'omniprésence des outils technologiques dans la vie actuelle. Ainsi, dans la suite de notre travail, nous désignons par serious game toute application informatique ayant pour objectif celui de faciliter l'apprentissage et l'enseignement, en utilisant des ressorts ludiques issus du jeu vidéo.

B. Stratégies d'utilisation des serious games en classe

Selon Kafai (2006), on peut distinguer entre deux approches lors de l'intégration des serious games en classe:

- Approche instructionniste : le serious game est considéré comme un outil pédagogique et didactique permettant de faciliter la transmission du savoir. Son rôle est donc similaire à celui des matériels pédagogiques et outils didactiques tels que les films documentaires, les livres, les articles de journaux pour les mêmes finalités. En effet, l'élève lors d'une activité encadrée par l'enseignant est invité à jouer un jeu qui à la fin lui enseigne un concept donné.

[6] "*It is the player and not the designer who decides how to use a toy, a game, or a videogame. The designer might suggest a set of rules, but the player has always the final decision*" page 14

Ce potentiel des serious games a été mis en valeur dans des champs multiples et variés, on peut citer par exemple : le jeu *supercharged* (Squire, 2004) pour enseigner l'électromagnétisme, le jeu *Zombie Division* (Habgood, 2005) pour enseigner les mathématiques, *Thélème* (Schmoll, 2011) et *Tactical Iraqi* (Johnson, 2007) pour développer et évaluer des compétences linguistiques et culturelles.

- Approche constructionniste : développée par Seymour Papert (1991). Elle est basée sur le fondement de la théorie constructiviste où l'apprentissage est une construction d'un savoir. Cette approche précise que l'apprentissage est particulièrement efficace dans un contexte où l'apprenant est consciemment engagé dans la construction de quelque chose. Dans ce sens, l'élève est un acteur dans le processus de la création du jeu. Ainsi, on lui attribue une grande liberté pour développer les compétences visées par l'enseignant avec son propre style d'apprentissage. Pour mettre en application sa théorie, Seymour Papert a développé l'environnement LOGO[7]. L'idée de base est de fournir un environnement simple dans lequel les enfants peuvent apprendre et communiquer avec l'ordinateur avec des instructions simples (Papert, 1980).
  La recherche bibliographique effectuée a permis de mettre en évidence plusieurs expérimentations réussies dans cette approche :
  - les travaux de Kafai sur l'apprentissage des fractions mathématiques utilisant l'environnement Logo (Kafai, 1995) ;
  - *Game Maker*, logiciel proposé pour apprendre la programmation (Overmars, 2004).

  Il faut noter que l'approche constructionniste s'inspire de la nature intrinsèque du constructivisme. En effet, elle met la création de jeux au cœur de l'activité pédagogique, ainsi l'apprenant est impliqué dans toutes les décisions de conception et en parallèle il développe des compétences très demandées pour vivre en 21ème siècle: le travail en groupe, le partage des connaissances, la communication, l'utilisation des outils technologiques …

Dans ce travail ayant pour objectif d'introduire l'algorithmique et la programmation aux élèves de la 1ère année du secondaire qualifiant au Maroc d'une façon innovante, nous avons adopté une approche qui s'inspire des deux approches ci-dessus : l'élève est invité à jouer à des serious games pour apprendre, puis il est amené à créer des jeux en utilisant un environnement visuel de programmation (Scratch). Avant de détailler notre méthodologie, nous allons présenter des outils permettant l'apprentissage des notions de base de l'algorithmique et de la programmation d'une façon ludique.

C. Outils pour apprendre la programmation et l'algorithmique d'une façon ludique

1. Apprentissage de l'algorithmique et de la programmation par la pratique des serious games

Un serious game peut être utilisé pour transmettre des concepts fondamentaux de la programmation. Dans ce cas chaque mission proposée dans le jeu peut être liée à une ou plusieurs notions spécifiques. Ainsi l'apprenant, lors de sa quête pour accomplir les tâches et surmonter les défis du jeu, acquiert des connaissances et développe des stratégies de programmation.

La plupart de ces jeux mettent l'accent sur les concepts sous-jacents de la programmation et la résolution de problèmes au lieu de se focaliser sur les spécificités de la syntaxe. Ces jeux sont privilégiés pour les débutants et permettent le plus souvent de construire des programmes à partir d'un ensemble de blocs de commandes prédéfinies pour piloter un robot ou un objet. Néanmoins certains jeux permettent d'entrer des séquences de commandes (Paliokas et al., 2011), de compléter des fonctions ou déboguer un code existant pour surmonter les obstacles et résoudre les tâches proposées dans le jeu avec une syntaxe spécifique à un langage de programmation (Barnes et al., 2007 ; Chaffin et al., 2009 ; Li et al., 2011).

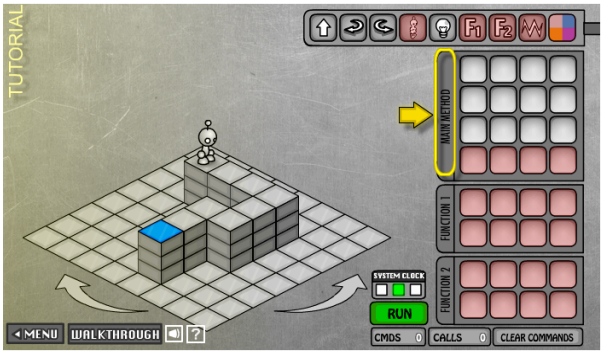
Nous présentons ci-dessous quelques exemples de serious games qui se focalisent sur l'apprentissage

[7] http://el.media.mit.edu/logo-foundation/index.html, consulté le 14/01/2016

des notions de base de la programmation sans tenir compte de la syntaxe.

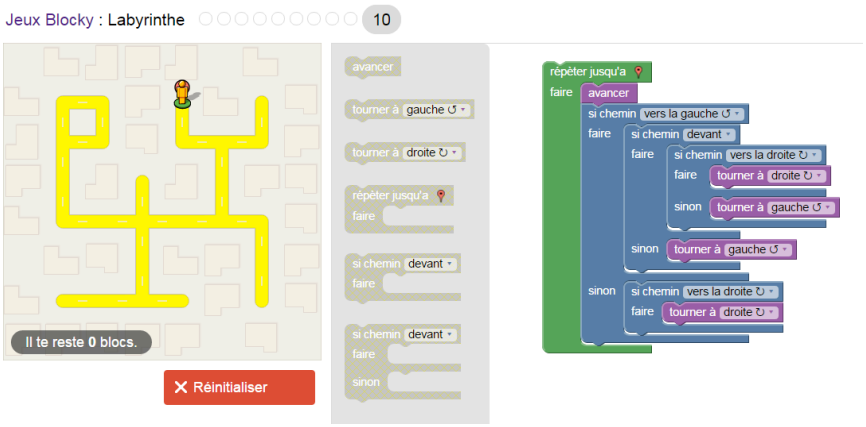
Light-Bot[8] *:* le joueur contrôle un robot virtuel à l'aide d'un langage visuel simple pour éclairer des carreaux (*figure 1*). Dans sa mission pour résoudre les défis proposés, les joueurs cultivent une réelle compréhension des procédures, des boucles et des instructions conditionnelles, qui sont parmi les concepts de base de l'algorithmique et de la programmation.

Figure 1. Exemple de mission dans "Light-Bot 2"



Jeux Blockly[9] *:* une série de jeux éducatifs développés dans le cadre d'un projet de *Google* pour encourager l'apprentissage de la programmation pour les débutants. Ces jeux peuvent se jouer en ligne ou être téléchargés. L'exemple de jeux Blockly "Labyrinthe" (*figure 2*) introduit les notions de boucles et de sélection en proposant des niveaux de jeu avec un degré de difficulté progressif.

Figure 2. Jeu de labyrinthe, un exemple de jeux Blockley



L'initiative "Une Heure de Code" : récemment, il y a eu des développements encourageants pour promouvoir l'enseignement de la programmation et la pensée informatique en classe (Repenning et al., 2016). Nous citons l'initiative "Une Heure de Code" "*Hour of Code*"[10], organisée par *Code.org* : une organisation à but non lucratif, dédiée à la démocratisation de l'informatique. Cette initiative représente le mouvement le plus large qui permet de créer de multiples expériences pour les jeunes apprenants dans le domaine de la programmation et la pensée informatique à travers le monde : plus de 430 millions[11] d'élèves ont essayé une heure de code dans plus de 180 pays. Le site *Code Studio*[12] héberge les cours et les tutoriels créés par *Code.org*, ces ressources sont souvent présentées sous forme d'activités ludiques, jeux et des séquences vidéo.

[8] http://armorgames.com/play/6061/light-bot-20, , consulté le 27/08/2016
[9] https://blockly-games.appspot.com/?lang=fr, , consulté le 27/08/2016
[10] http://code.org, consulté le 27/08/2016
[11] https://hourofcode.com/fr, consulté le 08/09/2017
[12] https://studio.code.org/, consulté le 30/08/2016

2. Aperçu sur les environnements visuels facilitant la programmation

a. Intérêt des environnements visuels de programmation

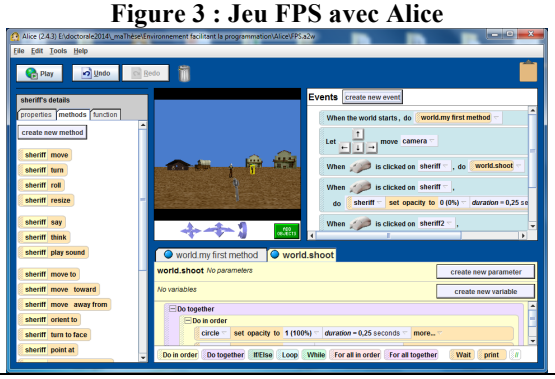
Plusieurs efforts ont été effectués pour rendre la programmation accessible aux débutants et d'introduire la programmation d'une manière plus facile (Mendelson et al., 1990). En effet, au fil des années un certain nombre d'environnements de programmation sont développés (McNerney, 2004; Kelleher et al., 2005) pour permettre aux débutants d'apprendre les bases de la programmation. Ces environnements sont spécialement conçus pour l'éducation, en effet :

- ils ont une syntaxe simple et réduite, ce qui réduit fortement le temps d'apprentissage des rudiments du langage, pour ainsi se consacrer au développement des algorithmes et à la compréhension des notions de base de la programmation : les instructions de base, les conditions, les boucles et la construction d'un programme simple ;
- ils sont visuellement attrayants : il est possible de créer des programmes qui couvrent les bases de la programmation et soient en relation avec les contextes tirés de la vie quotidienne des apprenants ;
- dans ces plateformes l'élève observe le résultat de ces manipulations et se concentre sur la façon d'organiser ces commandes selon une certaine logique pour résoudre un problème donné. Il a aussi la possibilité de tester les commandes ou s'amuser à les enchainer tout en recevant un feedback immédiatement ;
- la probabilité de commettre des erreurs de syntaxe est faible (Tempel, 2013), ce qui rend la programmation réconfortante pour les débutants ;
- les commandes et les mots clés peuvent être utilisés à l'aide de la langue maternelle des élèves ;
- ces environnements ne fournissent que des types de données et des structures de contrôle essentiels pour ces élèves ;
- ces environnements peuvent fournir une base solide pour apprendre les langages de programmation générale comme PASCAL, C, JAVA… ;
- en plus, quel que soit la filière de l'élève, son niveau d'études ou son âge, ces environnements peuvent enseigner des compétences technologiques aux apprenants, la logique algorithmique, ainsi que le travail en collaboration et la communication, qui sont des compétences fréquemment sollicitées au 21ème siècle.

b. Exemples d'environnements facilitant la programmation et la création vidéoludique

Le *Tableau I* ci-dessous présente des exemples d'environnements visuels permettant l'apprentissage des notions de base de l'algorithmique et de la programmation d'une façon ludique.

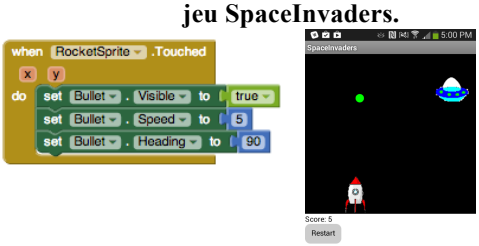
Tableau I. Exemples d'environnements visuels

Alice[13] est créé par un groupe de recherche dirigé par Randy Pausch à l'université Carnegie Mellon (Cooper et al., 2000). Il fournit un environnement (*figure 3*) dans lequel les élèves peuvent utiliser ou modifier des objets 3D. Il leur permet aussi d'écrire des programmes pour créer des animations interactives, jeux vidéo, et histoires. Alice peut être utilisé comme une première approche pour introduire la programmation orientée objet et faciliter l'apprentissage des langages de programmations générales comme JAVA et C++ au niveau universitaire (Allinjawi et al., 2014 ; Daly, 2011).	**Figure 3 : Jeu FPS avec Alice** 

[13] http://www.alice.org/, consulté le 31/01/2015

App Inventor[14] est basé sur les environnements Scratch et StarLogo TNG. Il fut développé par un groupe de chercheurs de *Google*[15]. Il rend accessible même pour les débutants la programmation et la création des applications fonctionnelles pour les appareils Android (*figure 4*). Avec cet environnement en plus de l'apprentissage des notions de base de la programmation, les apprenants peuvent créer des applications qui ont une utilité en situation réelle, ce qui permet de les motiver à s'engager dans la résolution des problèmes logiques plus complexes (Wolber, 2011).

Figure 4. Exemple de blocks App Inventor dans le jeu SpaceInvaders.



http://explore.appinventor.mit.edu/ai2/space-invaders

Scratch[16] (*figure 5*) a été développé par le groupe de recherche *Lifelong Kindergarten* auprès du laboratoire Média du MIT (*Massachusetts Institute of Technology*). Il facilite la création d'histoires interactives, de dessins animés, de jeux, de compositions musicales, de simulations numériques et leurs partages sur le Web. Le fait que Scratch est open-source a permis la création de plusieurs projets comme *BYOD*[17] (récemment nommé Snap!) et Panther[18]. Scratch présente aussi l'avantage d'être utilisé pour introduire la robotique pédagogique; par le pilotage et la programmation d'interfaces comme la carte Arduino et la carte d'acquisition PicoBoard.

Plusieurs travaux ont prouvé l'efficacité de Scratch dans l'enseignement des concepts de base de programmation, ainsi que l'acquisition des compétences de haut niveau comme la communication, le partage, la pensée critique, la résolution de problèmes, l'autonomie et la créativité chez les élèves du primaire et du secondaire (Romeike, 2007; Wilson et al., 2013; Calao et al., 2015). Scratch a fait ses preuves aussi au niveau universitaire (Malan et al., 2007; Baron et al., 2013) et dans la formation des enseignants (Kim et al., 2012; Ouahbi et al., 2016).

Figure 5. Exemple de script dans Scratch

III. Méthodologie et méthodes

Afin d'apporter des éléments de réponse à notre question de recherche qui concerne l'enseignement du module "Algorithmique et programmation" et ainsi vérifier notre hypothèse de recherche déjà mentionnée plus haut : *"l'usage des serious games et la création des jeux à l'aide d'un environnement facilitant la programmation tel que Scratch, favorise un apprentissage efficace du module algorithmique et programmation",* nous avons adopté la méthodologie décrite dans les paragraphes suivants :

Dans un premier temps nous avons souligné les compétences visées par le ministère de l'éducation nationale (MEN) et qui sont prescrites dans les instructions officielles qui concernent l'enseignement de la discipline informatique. Ensuite, nous avons élaboré des séquences pédagogiques se rapportant à ces compétences, basées sur la pratique et la création des serious games en utilisant l'environnement scratch. A la fin, pour évaluer l'impact de l'enseignement par les serious games nous avons comparé les acquis des élèves du groupe expérimental avec les élèves du groupe témoin qui ont suivi des cours standards. Ainsi les élèves des deux groupes ont passé un test d'évaluation qui entre dans le cadre des évaluations sommatives et ont répondu à un questionnaire de satisfaction.

[14] http://appinventor.mit.edu/, consulté le 30/01/2015
[15] http://googleresearch.blogspot.com/2009/07/app-inventor-for-android.html, consulté le 31/01/2015
[16] https://scratch.mit.edu/,
[17] http://byob.berkeley.edu/, consulté le 20/06/2015
[18] http://pantherprogramming.weebly.com/, consulté le 20/06/2015

A. Compétences visées

Nous rapportons dans le *Tableau II* ci-dessous les compétences visées et les savoirs associés qui concrétisent le contenu des séquences pédagogiques élaborées.

Tableau II. Compétences prescrites dans les instructions officielles du ministère concernant le module "Algorithmique et programmation" (MEN, 2005)

Algorithmique et programmation Module 3 :	
Compétences et capacités visées	**Savoirs associés**
L'apprenant doit être capable d'adopter la démarche algorithmique pour faire face à des situations problèmes.	➢ Constantes, variables et types ; ➢ Instructions de base (lecture, écriture, affectation) ; ➢ Structure de contrôle de base ; ▪ Séquentielle ; - Opérateurs algébriques ; - Représentation des algorithmes séquentiels. ▪ Sélective ; - Opérateurs rationnels et logiques ; - Structure sélective simple imbriquée à choix multiple ; - Représentation des algorithmes sélectifs. ➢ Langages de programmation ; ▪ Notion de programme (définition, exemples) ; ▪ Langages de programmation ; ▪ Transcription d'algorithmes. ➢ Notions d'algorithmes

B. L'environnement "Scratch"

L'environnement utilisé dans ce travail est Scratch (Resnick et al., 2009 ; Maloney et al., 2010). Cet environnement a connu une évolution considérable dans les dernières années, il dispose d'une communauté très active, ainsi que d'une très large bibliothèque de programmes qu'il est possible de reprendre et de modifier. Il compte plus de 20 millions utilisateurs inscrits et plus de 24 millions projets partagés[19]. Scratch est connu par sa vocation éducative et ludique, ce qui en fait un environnement optimal pour notre contexte.

C. Echantillonnage

L'échantillon se compose de 40 élèves de la 1ère année scientifique du secondaire qualifiant choisis de façon arbitraire. Ces élèves ont été répartis en deux groupes de 20 élèves chacun: groupe expérimental (GE) et groupe témoin (GT). Le groupe GE a bénéficié d'un enseignement basé sur les serious games alors que le GT a suivi un enseignement classique.

D. Déroulement de l'expérimentation

Les élèves du GE ont d'abord été initiés à exercer avec des serious games pour appréhender la démarche algorithmique, suivis d'exemples théoriques. Après cette initiation nous avons introduit les instructions de base de la programmation sous forme de travaux pratiques où l'élève est amené à créer des jeux avec l'environnement Scratch (*figure 6*) pour appliquer les notions de base de la programmation abordées dans le cours.

Les élèves du GT ont suivi des séances d'algorithmique en se basant sur des exemples théoriques. Les instructions de base de la programmation ont été introduites sous forme de travaux pratiques classiques : l'élève crée des programmes (généralement basés sur les mathématiques, sous la forme :

[19] http://scratch.mit.edu/ (consulté le 08/09/2017)

écrire un algorithme qui calcule …) en utilisant l'environnement Pascal.

Toutes les séances des TP ont été réalisées dans la salle multimédia de l'établissement, équipée de 10 ordinateurs avec accès Internet. Les séances pratiques sont réalisées par binôme. Les différentes activités réalisées sont décrites dans le *tableau III* ci-dessous :

Tableau III. Activités réalisées durant l'expérimentation

Items	Groupe expérimental	Groupe témoin
Comprendre la logique algorithmique	✓ Les élèves sont initiés et puis amenés à exercer avec des serious games : - jeux de logique : le loup et la chèvre[20], les petits chinois[21], le déplacement de grenouille[22], - jeux de labyrinthes Blockly, Light-Bot et des jeux disponibles sur *Code.org* ✓ Notion d'algorithme d'une façon classique	✓ Notion d'algorithme d'une façon classique
Instructions de base et structure de contrôle	✓ L'élève est invité à exercer avec le serious game : jeux de labyrinthes Blockly ✓ Exemples d'algorithme classique ✓ L'élève est amené à créer des serious games (labyrinthe, jeux de tir…) avec Scratch pour mettre en action les notions de base de la programmation.	✓ Exemples d'algorithme classique ✓ L'élève est amené à créer des programmes simples en utilisant l'environnement de programmation Pascal
Évaluation (voir la partie ci-dessous)	✓ Test de connaissance (phase 1) ✓ Questionnaire (phase 2)	

Figure 6. Activités ludiques avec l'environnement Scratch

[20] http://jeux.lulu.pagesperso-orange.fr/html/loupChe/loupChe1.htm, consulté le 25/02/2015
[21] http://www.logicieleducatif.fr/math/logique/chinois.php, consulté le 25/02/2015
[22] http://jeu.info/logique-grenouille.html, consulté le 25/02/2015

E. Traitement des données de l'expérimentation

Au terme de l'expérimentation les élèves ont été amenés d'une part à passer un test formel qui entre dans le cadre des évaluations sommatives et d'autre part à répondre à un questionnaire de diagnostic de la motivation. Ainsi, les données de l'expérimentation ont été traitées en deux phases.

a. Phase 1 : Évaluation des élèves

L'évaluation des élèves a porté sur les notions et compétences visées par les instructions officielles relatives au module algorithmique et programmation. Le test est structuré en deux parties :

- Partie I : ayant pour finalité d'évaluer les connaissances des élèves sur les notions de base de l'algorithmique.
- Partie II : permettant de mesurer leur capacité à appliquer la logique algorithmique pour créer des algorithmes et des programmes donnés.

Les données recueillies ont été codées et sont enregistrées sous forme d'une base de données. Leur traitement a été effectué en utilisant le logiciel SPSS pour Windows (version 18). L'analyse a consisté au test de Shapiro-Wilk, des représentations en graphe Q-Q et boîte à moustaches. Ainsi qu'une série de tests unilatéraux du test *student* appliqués pour des échantillons indépendants.

Le choix de recourir à ces tests a été fait pour comparer la pertinence de notre approche basée sur les serious games avec celle basée sur la méthode classique.

b. Phase 2 : Questionnaire de satisfaction

Afin d'avoir un aperçu sur la motivation à poursuivre des études en algorithmique et programmation, nous avons élaboré un questionnaire de satisfaction. Notre questionnaire est composé de deux parties :

- une partie profil s'articulant autour des informations personnelles des élèves : genre, âge, ainsi que les supports technologiques qu'ils utilisent.
- une 2ème partie constituée de 13 questions, les deux premières sont des questions fermées et les autres sont à échelle de Likert. Elle vise identifier les attitudes des élèves sur le module algorithmique et programmation :
    - les deux premières questions (1 et 2) renseignent sur l'expérience de l'élève avec l'environnement utilisé en classe : on veut savoir si l'élève a installé l'environnement de programmation utilisé à la maison, ainsi que le nombre d'heure d'utilisation,
    - les questions 3 à 7 informent sur l'avis des élèves sur la programmation à l'aide de l'environnement utilisé en classe,
    - les questions 8 à 12 permettent de connaître les attitudes des élèves sur la programmation.
    - la dernière question vise mesurer la motivation de l'élève à poursuivre des études postérieures en algorithmique et programmation.

Il faut noter qu'avant de répondre au questionnaire, le professeur a expliqué les questions posées et les termes clés utilisés : créativité, amusante, intéressante, coopératif,… Les données recueillies sont analysées à l'aide du logiciel SPHINX.

VI. Résultats

A. Résultat de l'évaluation des élèves (phase 1)

Les notes des élèves des deux groupes GE et GT obtenus lors de l'évaluation (phase 1) ont été analysées par les tests mentionnés ci-dessus.

1. Test de normalité

Les tests de normalités s'appuient sur des techniques empiriques des méthodes graphiques et des tests statistiques (Rakotomalala, 2011; Razali et al., 2011). Pour notre jeu de données on a utilisé :

- Les coefficients d'asymétrie "*skewness*" et d'aplatissement "*kurtosis*".
- La p-value du test de *Shapiro-Wilk*. En effet ce test, en comparaison avec d'autres tests statistiques est particulièrement puissant pour les petits effectifs inférieurs à 50 (Rakotomalala, 2011).
- La représentation en boîte à moustaches "*Box plot*" et en graphe quantile-quantile "*Q-Q plots*".

Les résultats du test de Shapiro-Wilk (*Tableau IV*; p-value>0.05), les représentations en boîte à moustaches et graphe Q-Q obtenu avec SPSS et les coefficients d'asymétrie et d'aplatissement (*Tableau V*) permettent de conclure que les données sont approximativement normalement distribuées.

Tableau IV : Test de normalité de Shapiro-Wilk et de Kolmogorov-Smirnov.

Normalité

	groupe	Kolmogorov-Smirnov			Shapiro-Wilk		
		Statistique	ddl	Signification	Statistique	ddl	Signification
noteTest	expérimental	,107	20	,200*	,951	20	,375
	témoin	,127	20	,200*	,953	20	,414

Tableau V : Coefficients d'asymétrie "skewness" et d'aplatissement "kurtosis"

Descriptives

groupe			Statistique	Erreur standard	Z-value
noteTest	expérimental	Moyenne	11,3250	,92713	
		Variance	17,191		
		Asymétrie	-,012	,512	-0,023
		Aplatissement	-,923	,992	-0,930
	témoin	Moyenne	8,1000	1,02122	
		Variance	20,858		
		Asymétrie	,506	,512	0,988
		Aplatissement	-,648	,992	-0,653

2. Test student concernant les notes de l'évaluation des élèves

Notre méthodologie d'analyse des données issues de l'évaluation sommative s'appuie sur des tests à hypothèses, où on doit émettre d'une part l'hypothèse nulle et l'hypothèse alternative. Ainsi dans notre travail, nous cherchions à vérifier les hypothèses suivantes relatives à la variable « noteTest» qui correspond aux notes des élèves durant l'évaluation :

- L'hypothèse nulle H0 selon laquelle les notes au contrôle du groupe expérimental ne sont pas différentes de celles du groupe témoin.
- L'hypothèse H1 selon laquelle les notes au contrôle du groupe expérimental sont différentes de celles du groupe témoin.

Après avoir présenté nos hypothèses, nous avons ensuite effectué le test *student* pour échantillons indépendants relative à la variable « noteTest » (*Tableau VI*). En effet, par ce test, nous voulons montrer si la différence entre les moyennes des deux groupes est significative ou non.

Tableau VI : Test *student* pour échantillons indépendants

		Test de Levene sur l'égalité des variances		Test-t pour égalité des moyennes						
		F	Sig.	t	ddl	**Sig. (bilatérale)**	Différence moyenne	Différence écart-type	Intervalle de confiance 95% de la différence	
									Inférieure	Supérieure
noteTest	Hypothèse de variances égales	,075	,785	-2,323	38	,026	-3,17500	1,36660	-5,94154	-,40846
	Hypothèse de variances inégales			-2,323	37,761	,026	-3,17500	1,36660	-5,94211	-,40789

La p-value=0.026 < 0.05, donc on conclut qu'il y a une différence significative entre les deux groupes, le groupe expérimental a bien acquis les notions de base du module algorithmique et programmation. Ainsi, l'hypothèse nulle H0 est rejetée, nous pouvons conclure que l'enseignement basé sur les serious games permet de produire un effet positif sur les résultats des élèves.

B. Résultat du questionnaire (phase 2)

La lecture des données du questionnaire met en évidence que la majorité des élèves possèdent un ordinateur à la maison (19 élèves pour le GE ; 18 élèves pour le GT). Ils sont familiers avec les outils technologiques. Cependant, le nombre des élèves, qui ont installé l'environnement de programmation utilisé dans l'expérimentation à la maison, dans le GE est largement supérieur à celui dans le GT : 17 élèves pour le GE (3 élèves ont utilisé l'environnement Scratch moins d'une heure par semaine, 7 élèves entre 1h à 2h par semaine, 3 élèves entre 2h à 3h et 4 élèves l'ont utilisé plus de 4h par semaine) contre 2 élèves pour le GT (seulement un élève a utilisé l'environnement pascal entre 1h à 2h par semaine).

Les données relatives aux attitudes envers la programmation et l'environnement utilisé en classe sont représentées dans les histogrammes ci-dessous.

Figure 7 : Attitudes des élèves envers la programmation avec l'environnement utilisé

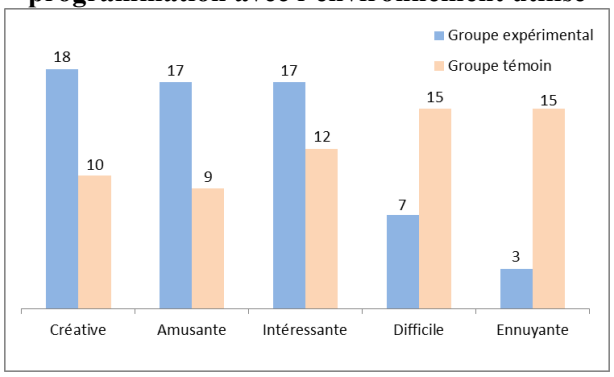
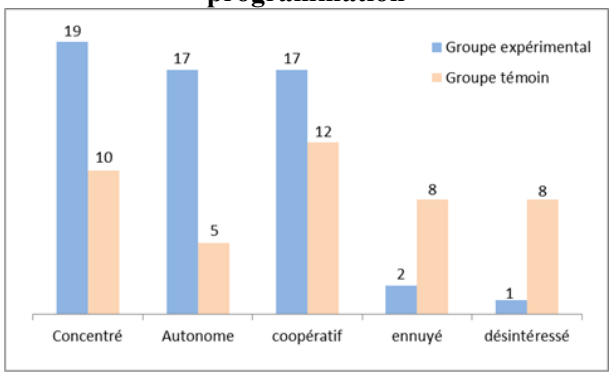


Figure 8 : Attitudes des élèves envers la programmation



Il ressort de ces histogrammes une différence très nette entre les attitudes des deux groupes GT et GE envers l'environnement utilisé. En effet, dans le GE environ 17 élèves trouvent la programmation à l'aide de l'environnement Scratch amusante, intéressante et créative. 7 élèves du GE trouvent la

création des programmes difficile avec Scratch contre 15 pour le GT (avec l'environnement Pascal). Bien que 3 élèves du GE trouvent la programmation avec le logiciel Scratch ennuyante, 15 élèves du GT trouvent la programmation avec un environnement standard ennuyante. La création des jeux et des histoires pour apprendre les notions de base de la programmation permet de rendre les élèves du GE plus autonomes, concentrés et coopératifs (environ 17 élèves). Lorsque les élèves ont été questionnés sur leur désir de poursuivre leurs études en programmation 13 élèves qui ont vécu l'expérience avec Scratch estiment poursuivre leurs études en programmation contre 3 élèves ayant utilisé un environnement standard de programmation.

On peut aussi souligner des réactions positives des élèves du GE : à la fin de chaque séance de travaux pratiques, ils applaudissent, se mettent à créer, se déplacent pour voir le travail des autres. Pour expliquer ces réactions, un élève a commenté : "*c'est génial, on est vraiment en train d'apprendre tout en s'amusant*». D'autres élèves ont présentés leurs propres réalisations aux autres élèves, il y avait vraiment un échange intéressant, et une dynamique entres les pairs chose absente dans le groupe témoin.

V. Conclusion

Dans cet article nous présentons la motivation et les résultats de notre innovation pédagogique basée sur les serious games pour l'enseignement du module algorithmique et programmation au secondaire qualifiant. Les élèves ont été initiés à l'utilisation et à la création des serious games avec un environnement visuel (Scratch). Une telle approche a permis aux élèves débutants d'implémenter des animations, des histoires, et des jeux. En plus, l'environnement Scratch leur a permis de se familiariser avec les fondamentaux de la programmation sans se soucier de la syntaxe et de les motiver à apprendre tout en s'amusant.

Les résultats de l'expérimentation montrent que l'apprentissage basé sur l'utilisation et la création des serious games pour l'apprentissage de l'algorithmique et de la programmation, peut être un moyen pour aboutir à un apprentissage efficace des élèves, les rendre plus autonomes dans la construction de leur savoir et compétences et de les motiver à poursuivre leurs études en programmation. Dans ce contexte nous proposons d'utiliser des environnements visuels à base de blocs comme Scratch et d'utiliser des serious games dans les cours d'introduction à l'algorithmique et à la programmation.

Références

Abt, C. C. (1970). *Serious Games*. New York : Viking Press.

Adams, E. (2010). The Designer's Notebook: Sorting Out the Genre Muddle. [En ligne] https://www.gamasutra.com/view/feature/4074/the_designers_notebook_sorting_

Ala-Mutka, K. (2012). Problems in learning and teaching programming. *Codewitz Needs Analysis*. [En ligne] https://www.cs.tut.fi/~edge/literature_study.pdf

Allinjawi, A. A., Al-Nuaim, H. A., & Krause, P. (2014). Evaluating the Effectiveness of a 3D Visualization Environment While Learning Object Oriented Programming. *Journal of Information Technology and Application in Education (JITAE)*. [En ligne] http://www.seipub.org/jitae/ paperInfo.aspx?ID=11989

Alvarez, J. (2007). *Du jeu vidéo au serious game : approches culturelle, pragmatique et formelle*. Thèse de doctorat soutenue le 17 décembre 2007, Université Toulouse 2 – Toulouse le Mirail / Université Toulous 3 – Paul Sabatier. [En ligne] http://ja.games.free.fr/These_SeriousGames/ TheseSeriousGames.pdf

Barker, L. J., McDowell, C., & Kalahar, K. (2009). Exploring factors that influence computer science introductory course students to persist in the major. *ACM SIGCSE Bulletin*, 41(1), 153-157.

Baron, G.-L., et Voulgre, E. (2013). Initier à la programmation des étudiants de master de sciences de l'éducation ? Un compte rendu d'expérience. Dans G.-L. Baron, É. Bruillard et B. Drot-Delange (dir.), *Sciences et technologies de l'information et de la communication en milieu éducatif : Objets et méthodes d'enseignement et d'apprentissage, de la maternelle à l'université*. Clermont-Ferrand : Université Blaise Pascal.

Barnes, T., Richter, H., Powell, E., Chaffin, A., & Godwin, A. (2007). Game2Learn : building CS1 learning games for retention. *ACM SIGCSE Bulletin*, 39(3), 121-125.

Bennedsen, J., Caspersen, M. E., & Kölling, M. (Eds.). (2008). *Reflections on the teaching of programming: methods and implementations* (Vol. 4821). Springer International Publishing.

Brito, M. A., & de Sá-Soares, F. (2014). Assessment frequency in introductory computer programming disciplines. *Computers in Human Behavior*, 30, 623-628.

Calao, L. A., Moreno-Le´on, J., Correa, H. E., & Robles, G. (2015). Developing Mathematical Thinking with Scratch. In G. Conole & T. Klobučar (Eds.), *Design for Teaching and Learning in a Networked World* (pp. 17-27). Springer International Publishing.

Chaffin, A., Doran, K., Hicks, D., & Barnes, T. (2009). Experimental evaluation of teaching recursion in a video game. In *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games* (pp. 79-86). ACM.

Crenshaw, T. L., Chambers, E. W. & Metcalf, H. (2008). A case study of retention practices at the University of Illinois at Urbana-Champaign. *ACM SIGCSE Bulletin*, 40(1), 412-416).

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges*, 15(5), 107-116.

Daly, T. (2011). Minimizing to maximize: an initial attempt at teaching introductory programming using Alice. *Journal of Computing Sciences in Colleges*, 26(5), 23-30.

Djaouti, D. (2011). *Serious Game Design: considérations théoriques et techniques sur la création de jeux vidéo à vocation utilitaire*. Thèse de doctorat soutenue le 28 novembre 2011, Université de Toulouse 3 Paul Sabatier. [En ligne] http://www.ludoscience.com/files/these_djaouti.pdf

Djaouti, D., Alvarez, J., Jessel, J. P., & Rampnoux, O. (2011). Origins of Serious Games. In M. Ma, A. Oikonomou, L. C. Jain (eds.), *Serious Games and edutainment applications* (pp. 25-43). London : Springer.

Frasca, G. (2003). Simulation versus Narrative: Introduction to Ludology. In M. J.P. Wolf & B. Perron (eds.), *The video game theory reader*. London : Routledge. [En ligne] http://www.ludology.org/articles/VGT_final.pdf

Freud, S. (1985). Le créateur littéraire et la fantaisie. In *L'inquiétante étrangeté et autres essais* (pp. 29-45). Paris : Gallimard.

Habgood, J. (2005). Zombie Division: intrinsic integration in digital learning games. *Cognitive Science Research Paper*, University of Sussex CSRP, 576, 45.

Henriot, J. (1989). *Sous couleur de jouer*. Paris : José Corti.

Johnson, W. L. (2007). Serious use of a serious game for language learning. *Frontiers in Artificial Intelligence and Applications*, 158, 67.

Ginat, D. (2004). On novice loop boundaries and range conceptions. *Computer Science Education*, 14(3), 165-181.

Kafai, Y. B. (1995). *Minds in Play: Computer Game Designs as a Context for Children's Learning*. London : Routledge.

Kafai, Y. B. (2006). Playing and making games for learning instructionist and constructionist perspectives for game studies. *Games and culture*, 1(1), 36-40.

Kasbi, Y. (2012). *Les Serious Games: une révolution*. Liège : Edipro.

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioral Sciences*, *47*, 1991-1999.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys* (*CSUR*), *37*(2), 83-137.

Kim, H., Choi, H., Han, J., & So, H. J. (2012). Enhancing teachers' ICT capacity for the 21st century learning environment: Three cases of teacher education in Korea. *Australasian Journal of Educational Technology*, *28*(6), 965-982.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, *37*(3), 14-18.

Li, F. W. B., & Watson, C. (2011). Game-based concept visualization for learning programming. In *Proceedings of the third international ACM workshop on Multimedia technologies for distance learning* (pp. 37-42). ACM.

Malan, D. J. & Leitner, H. H. (2007). Scratch for budding computer scientists. In *ACM SIGCSE Bulletin*, 39(1), 223-227.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, 10(4). [En ligne] http://web.media.mit.edu/~jmaloney/papers/ScratchLangAndEnvironment.pdf

Marfisi-Schottman, I. (2013). Méthodologie, modèles et outils pour la conception de Learning Games. Thèse de doctorat soutenue le 28 novembre 2012, Institut National des Sciences Appliquées (INSA) de Lyon. [En ligne] https://tel.archives-ouvertes.fr/tel-00762855v2

Matocha, J., Camp, T., & Hooper, R. (1998). Extended analogy: an alternative lecture method. *ACM SIGCSE Bulletin*, 30(1), 262-266.

McNerney, T. S. (2004). From turtles to Tangible Programming Bricks: explorations in physical language design. *Personal and Ubiquitous Computing*, 8(5), 326-337.

MEN (2005). *Programme et instructions officielles pour l'enseignement de l'informatique aux troncs communs*. Ministère d'Education National, de l'Enseignement Supérieur, de la formation des cadres et de la recherche scientifique, Département de l'Education Nationale, Secrétariat Général, Direction des curricula, Royaume du Maroc.

Mendelsohn, P., Green, T. R. G. & Brna, P. (1990). Programming languages in education: The search for an easy start. In Hoc, J-M., Green, T.R.G., Gilmore, D.J. and Samurcay, R. (Eds.) *The Psychology of Programming* (pp. 175-200). Academic Press.

Michael, D. R. & Chen, S. L. (2005). *Serious games: Games that educate, train, and inform*. Boston: Thomson Course Technology.

Michaud, L. & Alvarez, J. (2008). *Serious Games. Advergaming, edugaming, training and more*. Montpellier: IDATE Consulting & Research.

Muratet, M., Torguet, P., Jessel, J. P., &Viallet, F. (2009). Towards a serious game to help students learn computer programming. *International Journal of Computer Games Technology*, *2009*. [En ligne] https://www.hindawi.com/journals/ijcgt/2009/470590/

Mutet, S. (2003). *Simulation globale et formation des enseignants*. Gunter Narr Verlag.

Ouahbi, İ., Kaddari, F., Darhmaoui, H., Elachqar, A. & Lahmine, S (2014). Serious Games for teaching combined basic programming and English communication for non-science major students. *EduRe Journal*, 1, 77-89.

Ouahbi, I., Darhmaoui, H., Kaddari, F., Bemmouna, A., Elachqar, A. & Lahmine, S. (2015). Un aperçu sur l'enseignement de l'informatique au Maroc : Nécessité d'une réforme des curricula. *Frantice.net*, 11, 51-66. [En ligne] http://www.frantice.net/docannexe/fichier/1273/6.Ouahbi.pdf

Ouahbi, I., Darhmaoui, H., Kaddari, F., Elachqar, A. & Lahmine, S. (2016). Pre-service Teachers' Perceptions and Awareness toward Serious Games in the Classroom-Case of Morocco. In *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV)* (pp. 431-436). IEEE.

Overmars, M. (2004). Teaching computer science through game design. *Computer*, 37(4), 81-83.

Paliokas, I., Arapidis, C. & Mpimpitsos, M. (2011). PlayLOGO 3D: a 3D interactive video game for early programming education: let LOGO be a game. In *Games and Virtual Worlds for Serious Applications (VS-GAMES)*, 2011 Third International Conference on (pp. 24-31). IEEE.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

Papert, S. & Harel, I. (1991). Situating constructionism. *Constructionism*, 36, 1-11.

Rakotomalala, R. (2011). Tests de normalité: Techniques empiriques et tests statistiques, Version 2.0. Université Lumière Lyon 2. [En ligne] http://eric.univ-lyon2.fr/~ricco/cours/cours/Test_Normalite.pdf

Razali, N. M. & Wah, Y. B. (2011). Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of Statistical Modeling and Analytics*, 2(1), 21-33.

Repenning, A., Basawapatna, A., Assaf, D., Maiello, C. & Escherle, N. (2016). Retention of Flow: Evaluating a Computer Science Education Week Activity. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 633-638). ACM.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K. & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60-67.

Robins, A., Rountree, J. & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137-172.

Romeike, R. (2007). Applying creativity in CS high school education: criteria, teaching example and evaluation. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research*, vol. 88 (pp. 87-96). Australian Computer Society, Inc.

Sawyer, B. (2007). The "Serious Games" Landscape. Presented at *The Instructional & Research Technology Symposium for Arts*, Humanities and Social Sciences, Camden, USA.

Sawyer, B. & Rejeski, D. (2002). Serious Games: Improving public policy through game-based learning and simulation. [En ligne] https://fr.scribd.com/document/38259791/Serious-Games-Improving-Public-Policy-through-Gamebased-Learning-and-Simulation

Seppälä, O., Malmi, L. & Korhonen, A. (2006). Observations on student misconceptions - A case study of the Build–Heap Algorithm. *Computer Science Education*, 16(3), 241-255.

Schiller, F. (1943). *Lettres sur l'éducation esthétique de l'homme*. Paris : Aubier.

Schmoll, L. (2011). Usages éducatifs des jeux en ligne : l'exemple de l'apprentissage des langues. *Revue des sciences sociales*, *45*, 148-157.

Schmoll, P. (2013). Relire Jacques Henriot à l'ère de la société ludique et des jeux vidéo. *Sciences du jeu*, 1. [En ligne] http://sdj.revues.org/271

Schollmeyer, J. (2006). Games get serious. *Bulletin of the Atomic Scientists*, *62*, 34-39.

Squire, K., Barnett, M., Grant, J. M., & Higginbotham, T. (2004). Electromagnetism supercharged!: Learning physics with digital simulation games. In *Proceedings of the 6th international conference on Learning sciences* (pp. 513-520). International Society of the Learning Sciences.

Susi, T., Johannesson, M., & Backlund, P. (2007). Serious Games: An overview. Technical Report HS- IKI -TR-07-001, School of Humanities and Informatics, University of Skövde, Sweden. [En ligne] https://pdfs.semanticscholar.org/13e8/d4f8e2fe1bd2b82d63c0856c8585e15bb188.pdf

Tempel, M. (2013). Blocks Programming. *CSTA Voice*, 9(1). [En ligne] http://el.media.mit.edu/logo-foundation/resources/papers/pdf/blocks_programming.pdf

Wilson, A., Hainey, T., & Connolly, T. M. (2013). Using Scratch with Primary School Children: An Evaluation of Games Constructed to Gauge Understanding of Programming Concepts. *International Journal of Game-Based Learning (IJGBL), 3*(1), 93-109.

Wilson, C., Sudol, L. A. S., Stephenson, C., Stehlik, M., Acm, & Csta. (2011). Running on empty: The Failure to Teach K-12 Computer Science in the Digital Age. *Inquiry: A Journal of Medical Care Organization, Provision and Financing*, *48*(3), 177–82.

Wolber, D. (2011). App inventor and real-world motivation. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 601-606). SIGCSE 2011, Dallas, TX, USA. New-York: ACM.

Zyda, M. (2005). From visual simulation to virtual reality to games. *Computer*, *38*(9), 25-32.