A Serious Game for Learning of Variables and Operators Priority Rules in Programming

Chaker Abid, Hedia Mhiri Sellami and Lamjed Ben Said
abidchaker@yahoo.fr, hedia.mhiri@isg.rnu.tn, bensaid_lamjed@yahoo.fr
Laboratory of Strategies for Modeling and ARtificial intelligence
Institut Supérieur de Gestion de Tunis, Université de Tunis
Cité Bouchoucha 2000 Le Bardo,Tunis, Tunisia

Abstract— **The use of serious games has shown immense importance in several fields such as education and in particular the learning of computer programming. Several experiences have shown the positive impact of integrating serious games into programming learning. Some programming concepts present many difficulties for learners, especially for beginners, such as variables and operator priority rules. In this paper, we present the design, development and evaluation of a serious game called "AppProg Game" dedicated to learning these two programming concepts: variables and the priority of arithmetic and logical operators. The evaluation took place with a final class scientific section in a secondary school in Tunisia at the level of learning. The result of this experience showed that the integration of serious games helped learners to assimilate the two concepts discussed. The experiment showed that learners did not progress in the same way. For this reason, in our future experiment, we plan to adapt the game to the learners' profile using artificial intelligence techniques.**

I. Introduction

Learning to program has become an important skill for students to assimilate. Some studies have shown that it presents many difficulties for learners, especially beginners. These difficulties have led to a very high drop-out rate [1], [2] [1]. [3] reported that this rate varies from 25% to 80% for undergraduates in introductory programming courses worldwide. Therefore, [4] report that it is best for specialists in this field to focus on teaching methods to overcome these difficulties. The literature mentions several methods, such as the use of games. Games create conditions that encourage learning, such as interaction, feedback and the active participation of players. It helps develop skills in problem-solving, structuring and transposing knowledge, and promotes information reinforcement [5]. The use of games has gone beyond mere entertainment to achieve useful objectives known as serious games. A serious game is an artifact that combines the entertainment aspect of video games with a serious aspect to encourage players to achieve a serious objective. The uses of serious games in education or educational games have attracted specialists, because they are efficient and motivating environments that enhance learning [6]. The literature has mentioned many serious games used in different disciplines, such as history, languages, mathematics and programming learning. Integrating serious games into learning represents a promising pedagogical alternative. This article describes the design, development and evaluation of our serious game called "AppProg Game" dedicated to the learning of two programming concepts: variables and operator priority rules by secondary school students. These two concepts need to be well understood because we use them to read or write a program. In addition, many programming concepts are based on these two concepts, such as loops and functions. Also, the execution of a loop depends on conditions formed by variables and separated by logical operators. Programmers must be able to use arithmetic and logical operators in order to correctly evaluate an arithmetic or logical expression in order to understand the function of a code.

These two concepts are difficult to understand, especially for novice programmers. For example, variable names in programming are very important and must follow specifics rules. The name should represent the contents of the variable in a meaningful way to help the reader understand the program. Moreover, the priority of operators changes from one programming language to another and sometimes differs from mathematical rules. For these reasons, we have chosen to study them through this work. We aim to answer the following question:

Does the use of serious games help learners understand the two programming concepts mentioned?

The rest of this document is divided into 6 sections. In section 2, we identify programming learning difficulties and serious games dedicated to programming learning. In Section 3, we present the design and development of our game. Next, Section 4 describes the methods used, and Section 5 presents the application of the game and the analysis of the results gathered. Finally, we conclude with the outlook for this work.

II. State of the art

This section is divided into two parts. In the first part, we present some of the difficulties associated with learning programming, in particular variables and the precedence of operators. Then, in the second part, we present some serious games used for learning the two programming concepts studied at secondary school level.

A. Difficulties in Learning Variables and Operators Priority in Programming

The Learning to program presents many difficulties for learners, especially beginners, and even for teachers [7]. The literature mentions several difficulties that can be described as follows:

Difficulties related to problem-solving skills [8]: [7] have shown that the lack of these skills is explained, firstly, by difficulties in understanding the problem because they have misinterpreted the problem statement or they start writing the solution directly before they have fully understood the problem. Also, they have difficulty using their prior knowledge (knowledge transfer). Indeed, they can't make the analogy between the solutions of previously studied problems and the current problem.

Problems related to learners' conceptions: learners, especially beginners, have misconceptions that prevent them from understanding variables and operators priority. They consider that variables can contain several values at the same time [9]. According to [10], some students consider assignments to be symmetrical, e.g. y = 2 is the same as 2 = y. These misconceptions prevent them from understanding the code because they do not assimilate the changes in values of a variable during the execution of a code. According to [11], learners who do not master variables have difficulty assimilating other concepts such as loops. For example, they don't assimilate the automatic change of counters in loops. [12] mentioned that these misconceptions are a demotivating factor for learners.

Difficulties related to the variable: learners who do not have a perfect command of variables are unable to identify syntactical and logical errors. When an error occurs, they cannot find its meaning to correct it. Also, [13] showed that novice programmers do not name variables correctly, which can affect the quality of the program. In this context, [14] indicate that assigning more meaningful names to variables is more beneficial for code comprehension, debugging and program quality. [15], [16] have identified different roles for this concept: data (fixed value), counter (stepper), accumulator (gatherer), programming intermediary (temporary). For this reason, it can cause cognitive conflicts in secondary school learners.

Problems with teaching methods: the teacher focuses on teaching the programming language and syntax rather than the problem-solving approach. According to [17], programming consists of two phases: problem solving and code writing, and each phase involves specific skills. The problem-solving phase is the most important, as it forms the basis for the second phase, and teachers must give it priority. As indicated by [8], programming should be taught using personalized rather than traditional pedagogies. The teacher must monitor each learner individually and help him or her to solve his or her problem. This supervision is difficult to achieve because of time constraints and the content to be learned. The teacher must adopt the best teaching strategy according to several criteria, such as the skills of his or her students. [7] report that programming is a science that requires a high level of skills such as abstraction, generalization and critical thinking. In addition, programming languages can sometimes include a complicated syntax that is difficult to memorize. The learner must perform two difficult tasks: constructing the algorithm and mastering the syntactic rules of the programming language.

Difficulties linked to the precedence of operators: when evaluating an expression in programming, some students use mathematical rules. This can sometimes produce incorrect results, as the priority of operators is not the same in all programming languages.

B. Serious game for learning variables and operator priority in programming

Serious games are usually used to teach the two concepts studied simultaneously with other programming concepts, such as loops and functions. In this section, we present some serious games dedicated to learning programming that deal with these two concepts even implicitly. Some games allow learners to execute existing code to understand their function. For example, Robot ON! [18] a puzzle-style serious game for teaching programming to beginners, in which learners run existing programs to understand their purpose. It develops learners' critical thinking skills and helps them understand the problem-solving approach. Players use variables to store and retrieve values that represent many states in the game.

Another type of games provides incomplete codes and asks the learner to complete them using programming concepts such as variables and arithmetic and logical operators. Debugging is a difficult task, especially for beginners. In Code Hunt [19] and RoboBUG [20], the player runs programs and corrects errors to move from one level to the next. The player guides an avatar through the code to identify bugs and make the appropriate changes. A dialogue box appears at the bottom of the screen with comments on the debugging to be carried out. Players use variables to identify syntax and logic errors.

In a third kind of games, the player write code, and then debug it. Codecombat [21] is a puzzle game developed to teach programming concepts to beginners. It's made up of levels classified by degree of complexity. Players solve problems by writing code using variables and priority operators. He can write the complete program and execute it, or do it line by line. PlayLogo 3D [22] a competitive game to introduce children aged 6 to 13 to programming. Players control a robot, trying to fix their opponents' position and eliminate the gap between the two robots by writing LOGO commands. The use of variables to show the change in position values enables learners to understand the change in state of the variable in the code. The Gidget game [23] is a serious programming game that helps students understand debugging using a robot called Gidget. The player controls Gidget's programming by writing programs (list of commands) in a programming language that helps Gidget clean up the plant to liberate toxic substances and avoid the threat of animals. Gidget has limited energy, and if the player doesn't achieve the level objectives before Gidget runs out of energy, Gidget will fail and the player will have to try again. The use of the variable representing energy shows the change in state of the variable as the program is executed.

[24] proposed a serious game aimed at learning operator priority rules in evaluating an expression in the context of teaching introduction to programming. In this game, the player moves through a 3-dimensional maze, each time giving the solution to a given problem, which consists of evaluating an expression formed by operands and operators.

These serious games do not explicitly address the two concepts studied. They are used at the same time as other concepts, and the learner cannot easily assimilate them - for example, the counter in the loop. Also, many games use a list of commands to write or execute a program, rather than variables. [25] have shown that understanding of variable in programming is fundamental to manipulating loops, and that students have difficulty of understanding this concept. [26] considers the variable in programming to be didactically difficult. [27] has shown that this concept has several roles in the same program, depending on it's deployment. Moreover, learners do not always evaluate an expression correctly in introductory programming courses. They use the same rules as in mathematics, but these do not always apply to the programming language. [24] showed that in some programming languages expressions are evaluated according to operator precedence rules, while in others they are always evaluated from left to right.

These two concepts, variables and operator priority rules, are fundamental to programming and understanding them is necessary, especially for beginners. As mentioned above, we are going to use a serious game to help learners assimilate these two concepts. In the following section, we describe the design and development of this serious game.

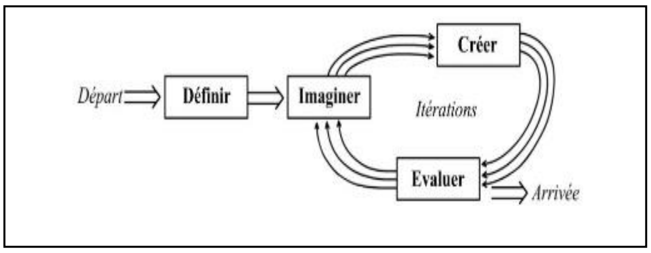
III. Design and develop a game "AppProg Game"

This section consists of two parts. We start with the design of our game then we describe it.

A. Game design

The design process of a serious game is a set of steps that describe the two dimensions of a serious game : playful and serious dimensions [28]. Literature has mentioned several models of serious game design. We chose to use the DICE design model (Fig. 1) composed of 4 steps [28]. Our choice is made with reference to the simplicity and clarity of this model and especially its iterative nature which allows us to make corrections as much as necessary [28]. The first step "Define" describes the specification of the serious content, which in our case consists of learning two programming concepts: variables and operator priority rules in secondary education. The second step "Imagine" describes how to combine the two playful and serious dimensions of play, which can be done in two ways: extrinsic and intrinsic [29]. In the intrinsic approach, the two dimensions are combined in such a way that they cannot be separated. The serious dimension is integrated with the playful dimension. We prefer the extrinsic approach, in which the two dimensions are separated and the game represents a reward following the course. The third "Create" stage concerns the creation of the game, with its two phases described below: design and development. In the development phase, we used the Python language. Finally, the fourth step "Evaluate" consists of evaluating the game to determine whether it meets expectations and to make any necessary corrections.

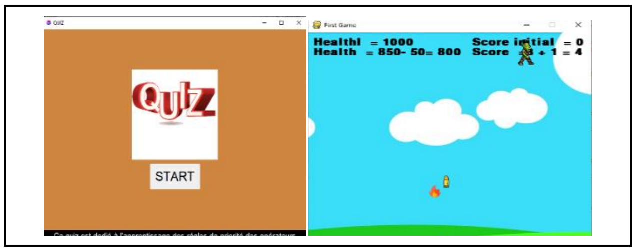
Figure 1. DICE Model [28]



B. Description of the game

AppProg Game contains two unordered levels (figure 2). The first level deals with the notion of variable (initialization, update and final value). It illustrates a fight between an avatar and enemies in a 2D scene. At the top of the screen, two variables are displayed, called score and health, to illustrate the player's score and power evolution respectively. They start from 0 and 1000 points respectively, to explain to the player the concept of initializing a variable. In the game, the learner moves an avatar left, right, forwards and backwards in the scene using the arrow keys on the keyboard, and fires bullets at enemies using the space key. When a bullet hits an enemy, the score increases by 1. Similarly, enemies also throw projectiles at the avatar. When a projectile hits the avatar, the health variable is decremented by 50 points. When the value of the health variable reaches 500 points, the text color of this variable turns red to inform the player of his critical state. We have chosen to display two additional texts containing the evolution of the content of the two variables: the increase of the score and the decrease of the health variable, so that the learner can follow the instantaneous evolution of the values of the two variables. The game ends when the player's score reaches 12, hence the player wins. A scene of joy is carried out to celebrate the event. However, when the health variable reaches 200, the game stops, and the player is loser. A scene of discontent is displayed.

Figure 2. Level 1 and 2 of the game



The second level contains a quiz consisting of a list of expressions to be evaluated by the player. Each expression consists of three operands separated by two operators (arithmetic or logical). When evaluating each expression, the learner must choose the correct answer from a list of choices accompanying the expression. Expressions are not ordered, and the player moves on to the next question, regardless of whether the answer is correct or not. When a correct answer is given, the score increases by 10 points. As the expressions are randomly generated, the learner can repeat the game as many times as he or she likes. At the end of the game, the player's score is displayed, along with a smile to express appreciation.

IV. Methods

We applied this game with a 4th grade sciences containing 78 students in a secondary school in Tunisia. We divided our sample into two groups of 39 students with heterogeneous levels in programming based on their results in a test carried out before starting the game. We applied the game with the first group of 28 boys and 11 girls, and performed the course in the usual way (without using the game) with the other group of 10 girls and 29 boys.

We carried out a pre-test and a post-test, each consisting of 10 questions on the same content, with both groups. Each test included 4 questions on variables and 6 questions on operator priority rules. The first two questions on variables focus on variable naming, which is an essential element to understand. The other two questions ask students to run a code to teach them how to initialise and change the state of a variable. The first two questions on operator precedence give learners expressions formed by operands and operators with a list of answers and the player has to choose the correct answer. The other questions give the player expressions and ask them to evaluate them. The generation of expressions is random in order to give the player the possibility of repeating the game several times.

The game took place over 8 two-hour sessions. In the first session, the learners completed a pre-test, then we presented them the game. We observed that the learners were very motivated and worked together to solve the puzzles. During the last session, the learners completed the post-test. In the other sessions, players played the game.

Evaluation took place before, during and after the game. Semi-structured interviews, observations and tests were used. In the game, the students were highly motivated and focused. In this work, we present the results of two tests that examine learners performance.

V. Discussions

The purpose of this research is to Closely scrutinise the effect of using the "AppProg game" to teach variables and operators priority rules in an introductory programming course for secondary school students. In the remainder of this paragraph, we consider that CA is a correct answer, IA an incorrect answer, G1 is group 1 and G2 group 2.

Table 1 shows learners' post-test and pre-test results. The results of the pre-test carried out before the game shows that the correct answers of the two groups were almost similar, reaching 51.28% in the first group (G1) and 52.56% in the second (G2). This can be explained by the fact that both groups were made up of learners with heterogeneous programming levels.

The post-test results show that the percentage of correct answers in the first group was 92.05% with an increase of 40.77%. In the second group, results rose from 52.56% in the pre-test to 67.18% in the post-test, an increase of 14.62%.

The results of both tests showed that the group that played the game performed better than the other group. The increase in their results was almost three times greater than that of the second group.

TABLE I. Percentages of the answers of the two groups in the post and pre test.ble Type Styles

	Pre-test		Post-test	
	Percentage of CA	***Percentage of IA***	***Percentage of CA***	***Percentage of IA***
G1	51.28%	48.72%	92.05%	7.95%
G2	52.56%	47.44%	67.18%	32.82%

We also studied the results of the two groups for each concept in Table 2.

Regarding variables, in the first group, correct answers rose from 49.35% in the pre-test to 92.30 in the post-test, an increase of 42.95%, while in the second group, correct answers rose from 49.35% in the pre-test to 69.87% in the post-test with an increase of 20.52%. The progression in results for learners in Group 1 is more than double that of Group 2 for this concept.

For the operator priority rules, the correct answers of Group 1 students rose from 52.56% in the pre-test to 91.88% in the post-test, an increase of 39.32%, while the correct answers of the second group rose from 54.70% in the pre-test to 65.38% in the post-test, an increase of 10.68%. Group 1 learners achieved more than three and a half times as much progress as Group 2 for this concept.

These results show that the learners in Group 1 assimilated both concepts better than their colleagues in Group 2. Consequently, we can say that the use of the game significantly helped learners in their learning of these concepts.

TABLE II. Results of the two groups in the two tests for each concept

		Pre-test				**Post-test**			
		G1		**G2**		**G1**		**G2**	
		CA	**IA**	**CA**	**IA**	**CA**	**IA**	**CA**	**IA**
Questions on the concept of variable	Q1	20	19	18	21	35	4	26	13
	Q2	22	17	23	16	32	7	27	12
	Q3	16	23	20	19	34	5	25	14
	Q4	19	20	18	21	30	9	24	15
Questions relating to the priority of operators	Q1	25	14	23	16	35	4	30	9
	Q2	20	19	21	18	30	9	27	12
	Q3	26	13	28	11	33	6	20	19
	Q4	21	19	19	20	34	5	25	14
	Q5	15	24	28	11	31	8	23	16
	Q6	16	23	16	23	29	10	20	19

We also looked at the distribution of post-test scores between the two groups, as shown in Table 3. In group 1, the number of students who obtained a score higher than 16 was five times higher than in group 2. In group 2, the percentage of students who obtained a score lower than 10 was 23.07% more than twice that of group 1, which was 10.25%. These results show that the integration of serious games not only enabled learners to better assimilate the two concepts covered, but also to achieve good results.

TABLE III. NUMBERS AND PERCENTAGES OF STUDENTS IN THE TWO GROUPS ACCORDING TO THEIR POST-TEST SCORES

	Marks <10		**Marks between 10 and 13,99**		**Marks between 14 and 16,99**		**Marks>16**	
	Nu mb er	***Percent ages***	***Num ber***	***Percent ages***	***Num ber***	***Percent ages***	***Num ber***	***Percent ages***
G1	4	10.25%	6	15%	14	36%	15	38,46%
G2	9	23.07%	20	51.28%	7	17,94%	3	8%

VI. CONCLUSION AND PERSPECTIVES

This article presents the design, development and evaluation of the "AppProg game" dedicated to learning variables and operator priority rules in computer programming. Analysis of the pre- and post-test results shows that the game helped learners to assimilate these concepts. This work is of interest to computer science teachers, educational engineers and computer science professors, particularly in secondary schools. It has limitations, such as the game's failure to take account of learners' characteristics and the lack of support for learners in difficulty, which can reduce their motivation. This is why, in future research, we plan to develop and evaluate an adaptive serious game for learning programming that adapts to players' levels and offers them support.

REFERENCES

[1] E. Lahtinen, K. Ala-Mutka, et H.-M. Järvinen, « A study of the difficulties of novice programmers », *Acm sigcse bulletin*, vol. 37, nº 3, p. 14-18, 2005.

[2] C. Watson et F. W. B. Li, « Failure rates in introductory programming revisited », in *Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE '14*, Uppsala, Sweden: ACM Press, 2014, p. 39-44. doi: 10.1145/2591708.2591749.

[3] J. Kaasboll, « Learning Programming », *University of Oslo*, 2002.

[4] A. Luxton-Reilly *et al.*, « Introductory programming: a systematic literature review », in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 2018, p. 55-106.

[5] L. Sauvé, L. Renaud, et M. Gauvin, « Une analyse des écrits sur les impacts du jeu sur l'apprentissage », *Revue des sciences de l'éducation*, vol. 33, nº 1, p. 89-107, 2007.

[6] D. Hooshyar, M. Yousefi, M. Wang, et H. Lim, « A data-driven procedural-content-generation approach for educational games », *Journal of Computer Assisted Learning*, vol. 34, nº 6, p. 731-739, 2018.

[7] S. A. Houssein et Y. Peter, « Outils d'assistance et les difficultés d'enseignement / apprentissage de la programmation, quelle aide ? », p. 8, 2017.

[8] A. Gomes et A. J. Mendes, « Learning to program-difficulties and solutions », in *International Conference on Engineering Education–ICEE*, 2007.

[9] F. Hermans, A. Swidan, E. Aivaloglou, et M. Smit, « Thinking out of the box: comparing metaphors for variables in programming education », in *Proceedings of the 13th Workshop in Primary and Secondary Computing Education*, Potsdam Germany: ACM, oct. 2018, p. 1-8. doi: 10.1145/3265757.3265765.

[10] T. Kohn, « Variable Evaluation: an Exploration of Novice Programmers' Understanding and Common Misconceptions », in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, Seattle Washington USA: ACM, mars 2017, p. 345-350. doi: 10.1145/3017680.3017724.

[11] A. Robins, J. Rountree, et N. Rountree, « Learning and teaching programming: A review and discussion », *Computer science education*, vol. 13, nº 2, p. 137-172, 2003.

[12] Y. Qian et J. Lehman, « Students' misconceptions and other difficulties in introductory programming: A literature review », *ACM Transactions on Computing Education (TOCE)*, vol. 18, nº 1, p. 1-24, 2017.

[13] A. R. M. Gobil, Z. Shukor, et I. A. Mohtar, « Novice difficulties in selection structure », in *2009 International Conference on Electrical Engineering and Informatics*, IEEE, 2009, p. 351-356. Consulté le: 5 décembre 2023. [En ligne]. Disponible sur: https://ieeexplore.ieee.org/abstract/document/5254715/

[14] E. Avidan et D. G. Feitelson, « Effects of variable names on comprehension: An empirical study », in *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, IEEE, 2017, p. 55-65. Consulté le: 5 décembre 2023. [En ligne]. Disponible sur: https://ieeexplore.ieee.org/abstract/document/7961504/

[15] J. Sajaniemi et R. Navarro-Prieto, « Roles of Variables in Experts' Programming Knowledge. », in *PPIG*, Citeseer, 2005, p. 13.

[16] J. Sajaniemi, « Guest Editor's Introduction: Psychology of Programming: Looking into Programmers Heads », *Human technology: an Interdisciplinary Journal on humans in ICT environments*, 2008.

[17] C. S. Cheah, « Factors contributing to the difficulties in teaching and learning of computer programming: A literature review », *Contemporary Educational Technology*, vol. 12, nº 2, p. ep272, 2020.

[18] M. A. Miljanovic et J. S. Bradbury, « Robot on! A serious game for improving programming comprehension », in *Proceedings of the 5th International Workshop on Games and Software Engineering*, 2016, p. 33-36.

[19] N. Tillmann, J. Bishop, N. Horspool, D. Perelman, et T. Xie, « Code hunt: searching for secret code for fun », in *Proceedings of the 7th International workshop on search-based software testing*, 2014, p. 23-26.

[20] M. A. Miljanovic et J. S. Bradbury, « Robobug: a serious game for learning debugging techniques », in *Proceedings of the 2017 acm conference on international computing education research*, 2017, p. 93-100.

[21] « CodeCombat - Coding games to learn Python and JavaScript ». Consulté le: 8 décembre 2022. [En ligne]. Disponible sur: https://codecombat.com/

[22] I. Paliokas, C. Arapidis, et M. Mpimpitsos, « Game based early programming education: the more you play, the more you learn », *Transactions on Edutainment IX*, p. 115-131, 2013.

[23] M. J. Lee, « Gidget: An online debugging game for learning and engagement in computing education », in *2014 ieee symposium on visual languages and human-centric computing (vl/hcc)*, IEEE, 2014, p. 193-194.

[24] N. Adamo-Villani, T. Haley-Hermiz, et R. Cutler, « Using a serious game approach to teach'operator precedence'to introductory programming students », in *2013 17th International Conference on Information Visualisation*, IEEE, 2013, p. 523-526.

[25] S. Grover, R. Pea, et S. Cooper, « Factors influencing computer science learning in middle school », in *Proceedings of the 47th ACM technical symposium on computing science education*, 2016, p. 552-557.

[26] É. Greff, « COMMENT PEUT-ON ABORDER LES PROBLÈMES DE PROGRAMMATION DANS LA NOUVELLE OPTION INFORMATIQUE DE SECONDE. ».

[27] J.-B. Lagrange et J. Rogalski, « Savoirs, concepts et situations dans les premiers apprentissages en programmation et en algorithmique », *Annales de Didactique et de Sciences Cognitives. Revue internationale de didactique des mathématiques*, nº 22, p. 119-158, 2017.

[28] D. Djaouti, « Serious Game Design: considérations théoriques et techniques sur la création de jeux vidéo à vocation utilitaire », PhD Thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2011.

[29] S. Egenfeldt-Nielsen, « Overview of research on the educational use of video games », *Nordic Journal of Digital Literacy*, vol. 1, nº 3, p. 184-214, 2006.