Educational Games for Teaching Computer Programming

Christos Malliarakis, Maya Satratzemi, and Stelios Xinogalos

Introduction

Students of the twenty-first century have been growing up in a highly digital world, where they learn and react very differently during the learning process in comparison to students not familiarised with technology. This is also due to the fact that computer games invade students' lives from a very early age, making them significantly accustomed to many of the computer's functionalities.

At the same time, students continue to face difficulties in computer science courses such as computer programming, even though their familiarisation with computers would suggest their learning of programming would become easier nowadays. The extended research studies carried out the last two decades state that these difficulties are still present and students seem to be even less interested in programming (Lahtinen, Ala-Mutka, & Jarvinen, 2005).

The hindering of these difficulties is important, as successful teaching and learning of computer programming can be extremely beneficial for twenty-first century generation students. It enables the development of various competences such as critical thinking, by allowing students to create their own programs, and of concept analysis and problem solving, as they are usually required to decipher a given scenario and interpret it into lines of code. Moreover, students learn to work in groups and collaborate with each other in their effort to develop executable programs while exercising in exchanging expertise and communicating ideas. Thus,

C. Malliarakis (✉)
Department of Applied Informatics, University of Macedonia,
156 Egnatia Street, 54006 Thessaloniki, Greece
e-mail: malliarakis@uom.gr

M. Satratzemi • S. Xinogalos
Department of Informatics, University of Macedonia,
156 Egnatia Street, 54006 Thessaloniki, Greece
e-mail: maya@uom.gr; stelios@uom.gr

overall computer programming empowers students to become lifelong learners, a very important benefit for this ever-growing world of knowledge since they can transfer their skills to a number of future work domains (engineering, computer science, etc.) (Law, Lee, & Yu, 2010).

An interesting proposal for alleviating the problems faced is the incorporation of educational games or serious games within computer programming courses. The term educational games is commonly used to describe computer games that are used as educational tools and provide interactive and appealing activities that attract students' interest for learning (Gunter, Kenny, & Vick, 2008). These games reinforce students' intrinsic motivation through the sense of challenge; they pique their curiosity, enforce a sense of security as well as stimulate their imagination (Ho, Chung, & Tsai, 2006). Also, students are able to achieve specific goals and view the success results immediately, not only when they complete the game but also when they pass the game's stages, a process that increases their self-confidence and helps them trust their decision making skills. Hence, there is a need to readjust currently followed learning techniques according to these newly emerged technological trends that can more easily and efficiently pull the new generation of students towards computer programming education.

The main purpose of this paper is to review existing educational games that aim to teach computer programming and correspondingly review how effectively they support students in achieving the educational goals that teachers set in each case. The rest of the paper is organised as follows: the next section provides information regarding the requirements that have been identified in the literature regarding the development of educational games. The following section presents a review and comparative analysis of educational games for teaching computer programming elements, giving emphasis in features that correspond to the requirements identified in the previous section. Finally, the paper concludes with an overview of the work done in the field in terms of how well and to what extend the studied educational games cover the identified requirements and suggests future work.

Educational Games Requirements Specification

Our work was carried out following a rigorous review methodology, where we searched a variety of academic databases (e.g. Web of Science, Scopus, CiteSeer and Google Scholar) for identifying relevant papers. During this search, we used keywords such as: educational games, computer programming, requirements, facilitating tools, etc., in various combinations. Over 70 papers were originally identified and in the end 20 were used for our review after thorough filtering as the most relevant to our scope of interest.

We have studied thoroughly case studies that have proposed and developed frameworks for educational games (Becker, 2010; De Freitas, & Jarvis, 2006; Salen & Zimmerman, 2004; Yusoff, Crowder, Gilbert, & Wills, 2009; Zualkernan, 2006). According to these works, the development of an educational game should be carried out after the examination of a number of aspects, where each aspect determines the features that should be supported in an educational game. All frameworks include similar requirements as concepts that are considered important. In this work, we choose to follow the one suggested by Becker (2010), because its concepts encompass all the ones included in previous frameworks. Thus, we consider it to be a more abstract superset of features that should be supported by all educational games.

Initially, it is suggested that the educational goals should be investigated across two axes by using a different viewpoint, so that their specification will be complete. These axes are:

- *Cognitive axis*, relating to mental competencies (Knowledge). Educational goals should make sure that the information received by the students begin from the first category in the Bloom's taxonomy (Knowledge) and end in the final and most complex category (Evaluation) successfully.
- *Emotional axis*, relating to emotions or emotional areas (Attitude). Educational goals should enable students to handle given situations through the enticement of their emotions. For example, the need and thus the desire to free a prisoner during a game motivates students to solve the assigned task faster and correctly so that they can experience the emotions followed by accomplishing the goal.

Additionally, it is important to select a proper framework that will guide the learning experience with the incorporation of educational games. The construction of a development framework requires the determination of various elements that together structure the framework's components. For example, the processes of the real world that will need to be simulated into the game (which movements will be allowed, how will the virtual world be constructed, how will the players be represented, etc.) need to be clearly identified. This is a very important step as it determines the educational scenarios that can be supported by the game's environment and thus affects the entire learning process.

Following the framework's specification, an architecture will be constructed based on the components identified that will have to be available in an authentic game environment. These components include:

- *The scenario's space*. Students are introduced to the game's storyline once they logon to the environment with a short description of the plot as well as a brief overview of the basic activities they will have to execute.
- *Relevant cases*. Students are provided with a set of pre-solved similar cases from which they can get a better insight of the game's knowledge and skills requirements and thus be better prepared for when it is their turn to solve assigned tasks.
- *Information resources*. Students can access information relevant to the task at hand whenever they are in need of assistance.
- *Facilitating tools*. A set of tools is included that students can use when they are trying to execute a task and that help them build new knowledge. Additionally, the game provides tools that underpin student communication as well as discussion with the teachers regarding any questions, thoughts or reflection on the virtual world.

Also, it is essential to distinguish *information* regarding the student, such as learning goals, learning style (holistic, analytical, etc.) as well cognitive limitations (e.g. behavioural competences that can affect their learning).

Teachers should also be able to set *educational goals* that will have to be accomplished by the students during the game by assigning specific activities for them to participate in. This way, students achieve interim goals by successfully completing tasks that will lead them to absorbing the final learning outcomes set by the teachers.

The above features will underpin the selection of an *authentic scenario* that will provide an attractive story to go along with the game's virtual world. To this end, students should be presented with an interesting and motivating problem that needs solving and it is best if the plot is similar to the ones available in existing computer games. This way, students will be already familiar with the overall concept and the activities they will engage in will seem more like games than teaching exercises. Similarly, the individual *problems* that students will be called to solve during the game should be consistent with the educational goals as well as with any cognitive limitations that may be apparent to the teachers. This is one more reason why both these features are required to be supported in an educational game.

Another important feature is the constant and explanatory *feedback* provided to the students during their navigation through the game's levels. This feedback should be represented in a form of messages that guide students towards understanding what they did right, what they did wrong and how they can achieve their goals, even through the mistakes they have made. This scaffolding technique ensures that students realise why their actions did not lead to successful task execution and thus they will be able to perform better in their next endeavour.

Finally, a number of *generic conditions* need to be taken into consideration while designing and developing an educational game. Such is the location and type of the education to take place (e.g. offline, online, blended learning). For example, if the learning process will be realised entirely online, then the game requires all the educational material to be uploaded within the environment and communication tools should be very well supported and plentiful. Moreover, the course's duration will determine how many hours students will spend on the game and how many scenarios need to be constructed according to the elements that need to be taught in this duration. All of the above require adequate preparation from the teacher and proper configuration of the environment so as to exploit all of the game's benefits and foster knowledge and skills development by the students.

Educational Games for Computer Programming Education

This section presents a series of games that have been developed specifically for computer programming courses. The review of these games was carried out based on the specifications identified and described in the previous section, in the cases where they were explicitly identified by the relevant literature. Moreover, all chosen games have addressed both the cognitive axis and the emotional axis during their development. Thus, in each game students start out by receiving pieces of information, and through their engagement with the game, they move on up the Bloom's taxonomy to the evaluation step by reflecting on their progress and finalising assignments. Also, the emotional axis is addressed via the game scenarios. All scenarios stimulate emotions that motivate students to go through all tasks in order to win.

Two major categories could be distinguished during the research, which sort educational games based on the educational goals they aim to support. The first category includes games that focus on teaching specific computer programming units while the second category represents games that cover multiple educational goals and thus computer programming material. A review for each category is presented in the next two subsections, followed by a comparative analysis presented in Table 1.

Educational Games Focused on Teaching a Specific Unit of Learning

Catacombs. It is a three-dimensional multiplayer game that aims to teach students how to declare variables and use simple and nested if statements and loops. According to the game's scenario, each player represents a wizard that has to rescue two children trapped within catacombs. Towards this goal, the wizards have to answer multiple choice questions trying to solve a given programming code that will help them complete their quests. The answers to the given questions automatically create executable lines of code in a micro-language. If the answers are correct, the wizards progress through the game's levels; otherwise they are given corresponding feedback as to what they answered wrong and are prompted to try again. The game records experience scores for each student and provides explanatory messages as a scaffolding mechanism (Barnes, Chaffin, Powell, & Lipford, 2008; Barnes et al., 2007).

Saving Princess Sera. It is a two-dimensional game that enables students' scaffolding through explanatory messages directed to the player. Each player has to try and save a princess named Sera who has been abducted by a monster named Gargamel, on her sixteenth birthday. Students are required to complete a number of quests in order to progress in the plot of the game. Towards this goal, they complete lines of code that will result to an executable program or they have to correctly map existing lines of code to their proper position or order within a program employing a drag and drop functionality. This way, students learn the quick-sort algorithm along with simple and nested loops with the usage of a micro-language (Barnes et al., 2007; Barnes et al., 2008).

EleMental: *The Recurrence*. It is a three-dimensional game that aims to teach students how to execute recursion and depth-first search transversal using the C# programming language. The player has to navigate across a virtual binary tree by employing the depth-first transversal and complete three quests by applying recursion. Two avatars named Ele and Cera help students during the game in various ways. For example, once the code is written, Ele crosses the binary tree according to how the written code is deployed, while Cera explains exactly what the code is producing at a specific moment (Chaffin, Doran, Hicks, & Barnes, 2009).

Wu's Castle. It is a two-dimensional role playing game that aims to teach students loops and arrays through interactive activities. Each player is a wizard that can control an army of snowmen. Players recognise logical errors at lines of code written in the C++ programming language. The game allows arrays management through changing the parameters inside the loops and movement of the characters through the execution of nested loops (Eagle & Barnes, 2009).

Robozzle. It is an online puzzle game that provides a series of predefined commands ready for use and does not show any actual code. According to the game's scenario, users have to build functions that will help them achieve each given task in a grid and tiles virtual world. Users can run their functions and see how their hero will move across the world and can therefore easily detect what mistakes they have made and reprogram accordingly (Li & Watson, 2011).

LightBot. LightBot (Piteira & Haddad, 2011) is an online puzzle game similar to Robozzle. It includes a series of predefined commands and no actual code or programming language. Additionally, users have to complete given tasks by building their own functions and moving the hero across a grid and tiles environment and light all the blue tiles. Once a task is completed, the user can move to the next level, which requires the construction of more complex functions.

TALENT. TALENT (Maragos & Grigoriadou, 2011) focuses on teaching if statements and loops in the forms of algorithms by using a micro-language. Each player is an archaeologist that has to navigate across the virtual environment by completing a series of tasks and collect objects that are available at specific locations for their future exhibition at a museum. Towards this goal, students can drag and drop lines of code as well as write them in an editor whenever requested. As a scaffolding mechanism, TALENT provides an agent that acts as a mentor and helps students when needed as well as suggest what their next mission should be.

Educational Games Focused on Teaching Multiple Units of Learning

Robocode. It is a two-dimensional environment that aims to teach computer programming using the Java language. The game comprises of a programming editor, robots and a virtual arena, and students are required to program a robot that will compete against one another in the arena. Students familiarise themselves with the basic commands of structured computer programming and object-oriented programming (e.g. inheritance, polymorphism) while they try to build a robot ready for combat. During its construction, the robot inherits basic methods that can later be extended by students according to the behaviour they want their robots to have inside the arena (O'Kelly & Gibson, 2006).

M.U.P.P.E.T.S. It is a three-dimensional, Web-based and collaborative game that aims to teach object building and in general to familiarise students with the basic concepts of object-oriented programming using the Java language. Students create a robot that has to fight another robot inside a virtual arena, interact with the objects they build and write and compile their lines of code within the embedded development environment that includes a commands console (Phelps, Bierre, & Parks, 2003).

Table 1 Overview of the educational games for computer programming courses

Game	Programming elements	Programming language	Programming activities	Special characteristics
Catacombs	Variables; simple and nested if statements; loops	Micro-language	Multiple choice questions; filling out lines of code	Three dimensional; multiplayer; success scores; scaffolding with explanatory messages from the hero
Saving Sera	If statements; recursion	Micro-language	Filling out lines of code; mapping parts of code in corresponding locations; multiple choice questions	Two dimensional; scaffolding
EleMental	Recursion; depth-first search (DFS) algorithm	C#	Depth-first search algorithm; moving the hero on a fantastical binary tree	Three dimensional; scaffolding
Prog&Play	Structured computer programming, if-statements, loops	Ada, C, Java, OCaml, Scratch, Compalgo	Completing eight missions	Multiplayer; infinite scenarios; available
Wu's Castle	Loops; arrays	C++	Arrays management; movement of the hero; identification of logical mistakes in code	Interaction; role playing
Robozzle	Functions	No code used	Creating functions through the hero's movement	Interactive; available
LightBot	Functions	No code used	Creating functions through the hero's movement	Web based; available
TALENT	If statements; loops	Micro-language	drag and drop lines of code; writing lines of code in an editor	Explanatory messages
Robocode	Structured computer programming; object-oriented programming	Java	Writing lines of code	Every robot stores data of past activities; available
M.U.P.P.E.T.S.	Three-dimensional objects; object-oriented programming	Java	Interaction with the developed objects; writing lines of code; compile	Collaborative; three dimensional; commands panel; embedded development environment
PlayLogo 3D	Basic concepts and commands of structured programming	Logo	Creation of heroes; navigation across the "galaxy" by writing commands	Interaction; multiplayer; three dimensional; available
Gidget	Analysis and design of basic algorithms	Micro-language	Fixing problems and programs; interaction with a personalised robot	Web based; explanatory messages

Prog&Play. It is a library currently integrated in the Web-based, real-time multiplayer strategy game Kernel Panic, which enables constant interaction amongst users. Students can program their own heroes and form alliances with each other aiming to prevail in the game. Prog&Play allows students to choose the language in which they prefer to code their programs amongst programming languages such as Ada, C, Java, OCaml, Scratch and Compalgo (Muratet, Torguet, Viallet, & Jessel, 2011).

PlayLogo 3D. It is a three-dimensional, role playing game that allows interaction amongst multiple users and aims to teach basic concepts of structured computer programming. Users are required to program their heroes by writing the corresponding lines of code in the LOGO language, and navigate them across the environment. More specifically, the virtual world consists of the spaceship X-15 located on a constellation of the Andromeda galaxy, where a contest is held each year amongst pilot-robots (Paliokas, Arapidis, & Mpimpitsos, 2011).

Gidget. It is a Web-based game where students can program using a simplified programming language created specifically for the game in order to learn how to design and analyse basic algorithms. A robot named Gidget has problems with a part of his software and thus cannot complete its tasks. Therefore, students are called in to help Gidget by either fixing wrong lines of code, or by completing missing code within given programs. During these processes, students receive constant feedback of their progress (Lee & Ko, 2011).

The above table presents the study with a structured representation of features supported by the most commonly known educational games for computer programming. The programming elements, characteristics as well programming activities identified can be considered as concepts that describe the field. Thus, future researchers and game developers should take them into consideration when designing a new and advanced educational game for computer programming.

Discussion

In this section we provide an overview of the development of existing educational games for computer programming, and their limitations. The results are categorised based on the features identified that should be taken into consideration during the design and development of an educational game for computer programming.

Educational goals. The educational goals seem to cover both the *cognitive* and *emotional axes*. Within the games, these goals are clearly focused in the computer programming concepts that each game aims to teach. This is especially the case in the educational games that cover specific units of learning, and thus the desired learning outcomes are more clearly identified. The emotional goals seem to be accomplished through the numerous attractive scenarios available in each game.

The *problems* students are required to solve are consistent with the set educational goals and their cognitive limitations. In the educational games focused in specific units of learning, students execute and complete quests that teach them knowledge that is relevant to the programming concepts set in the goals. As an example, the simple and nested loops in Catacombs are taught through the completion of lines of code, and their correct syntax allows students to pass to the next level, while the same concepts are taught in Wu's Castle when students move their characters across the world and recognise logical errors. On the other hand, the second category of educational games (e.g. Robocode, M.U.P.P.E.T.S, PlayLogo 3D) employs problems that allow students to interact with each other and execute multiple tasks that will teach them all the basic concepts of computer programming.

Framework. Educational games that focus on specific units of learning seem to have properly defined a framework for their employment in educational contexts. However, games that teach multiple and more complex units of learning, and thus cover multiple educational goals usually set several frameworks. It should be noted that the games Lightbot and Robozzle do not define any framework.

Scenario's space. All educational games present and work based on a *scenario* in order to attract and motivate students. In some cases *introductory information* is provided to the players in regard to the virtual world (e.g. PlayLogo 3D, Wu's Castle).

Information resources. Most educational games provide *explanatory messages*. The games where this feature is more fully supported are Catacombs, Saving Sera, EleMental: The Recurrence and Wu's Castle. Moreover, *scaffolding techniques* are provided through these explanatory messages that appear while students are trying to solve their quests (e.g. Catacombs, Saving Sera, EleMental).

Facilitating tools. Tools where students can write requested lines of code exist in the Catacombs, Saving Sera, EleMental, Robocode and M.U.P.P.E.T.S games. In addition, multiplayer games (e.g. PlayLogo 3D, M.U.P.P.E.T.S., Prog&Play, Catacombs) include features where students can *communicate and interact* with one another.

Generic conditions. The generic conditions have been taken into consideration. This has been carried out more efficiently in the educational games that cover specific units of learning rather than in the ones that teach multiple and complex computer programming concepts. On the other hand, they have not been considered at all during the design of the Robozzle and Lightbot games.

It should be noted that none of the studied games provides *relevant cases* that can prepare students for the activities they will be required to execute. The existence of this feature would significantly increase the quality of the games, since it would provide useful tutorials and guidelines for learners.

We also have to mention that many of the aforementioned information regarding these games derive exclusively from the relevant literature, since they are not available for access. This fact also results in our inability to exploit them in the learning process and actually test them against set educational goals in computer programming courses. Summing up, it seems that all studied games include scenarios that motivate learners, clearly indicate the educational goals that need to be reached and include problems that are set up as specified above. Other features, such as facilitating tools, information resources, one or several frameworks and taking into consideration generic conditions are supported by the majority of the studied games. However, none of the games appear to support relevant cases to prepare learners before engaging with the environments or to act as manuals for when learners require guidance.

Conclusions

The main implications derived from the analysis include that most games have been developed to cover programming concepts (such as variables, simple and nested if statements, loops, arrays, functions) with the exception of M.U.P.P.E.T.S. and Robocode that cover more complex concepts such as object-oriented programming. We do not consider the fact that the games do not tackle all programming concepts as a disadvantage, since they seem to successfully fulfil the educational goals they set regarding the group of concepts they aim to teach.

Also, the study elaborated on the added values of using educational games in computer programming. Our research revealed a number of interesting principles that can help us understand why educational games can improve teaching and learning of computer programming. For example, games seem to have a facilitating role in the learning process during the teaching of specified concepts and could play a small or a big part in the entire course's implementation process, depending on the generic conditions. More specifically, depending on the nature of the course (online, offline, blended), materials, communications, exams, etc. could be supported on different levels by the games' environments. To this end, educational games can provide a number of characteristics, such as storytelling, scaffolding and interactivity, which increase motivation for participation in class as well as attract students to complete their tasks through interesting scenarios.

We examined the educational games in terms of the educational value that they bring, and we derived that they can provide students with:

- Clear educational goals and learning outputs, ensuring that they know what they have to do to achieve the required knowledge and skills.
- A familiar and immersive environment that attracts students' attention facilitates their active participation and increases their motivation.
- Interesting scenarios with comprehensive problems they have to solve, which enable them to learn in a contextual manner (learning specific units of learning periodically).
- Tools that help them communicate and collaborate with their classmates, improving their group work skills and guiding them through the learning process by explaining the possible mistakes they make. These tools can be applied either with chat functionalities or with different types of interactions between the learners and the game while trying to achieve and fulfil common goals.

Furthermore, the study's findings have implications regarding the design of future educational games focused on computer programming, listing and elaborating on the requirements educational game designers and developers should strive to support and thus setting the foundations for future holistic environments.

Educational games can also assist teachers teach programming in their courses by designing the game and setting up its parameters. For example, teachers can use educational games to plan their courses and monitor students' interactions, progress and evaluation through their activities in the game. The establishment of the educational goals, learning outcomes and the setting up of a scenario that will delineate the curriculum into units of learning also enables teachers to be better prepared and have a deeper knowledge of the materials they teach and get more skilled in course planning.

On the other hand, a significant limitation identified in the existing educational games focused on computer programming courses is the ability of the teacher to configure the environment according to the pedagogical goals related to the respective unit of learning. Additionally, the collaboration concept could be reinforced and better supported within multiplayer educational games so that they can teach more complex programming concepts that will be more efficiently understood through team-based learning activities.

The evaluations carried out during the pilots studies, showed that the majority of learners expressed positive attitudes towards the examined environments. Thus, this enables the initial implications of our research to exploit the features considered important by the literature and presented throughout the paper during future design and development of educational games.

Such games will fully support all identified specifications and features and will aim to teach in-depth more complex concepts, such as object-oriented programming.

References

Barnes, T., Chaffin, A., Powell, E., & Lipford, H. (2008). Game2Learn: improving the motivation of CS1 students. *Proceedings of the 3rd International Conference on Game Development in Computer Science Education* (pp. 1–5). Miami, Florida.

Barnes, T., Richter, H., Chaffin, A., Godwin, A., Powell, E., Ralph, T., et al. (2007). The role of feedback in Game2Learn. *CHI, 2007*, 1–5.

Becker, T. (2010). The character of successful trainings with serious games. *International Journal Of Emerging Technologies In Learning (IJET)*, *5*(SI3). Retrieved April 17, 2012, from http://online-journals.org/i-jet/article/view/1498.

Chaffin, A., Doran, K., Hicks, D., & Barnes, T. (2009). Experimental evaluation of teaching recursion in a video game. In S. N. Spencer (Ed.), *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games* (New Orleans, Louisiana, August 04–06, 2009). *Sandbox '09* (pp. 79–86). New York, NY: ACM.

De Freitas, S., & Jarvis, S. (2006). A framework for developing serious games to meet learner needs. In *Proceedings Interservice/Industry Training, Simulation, and Education Conference, Florida, USA* (pp. 1–11).

Eagle, M., & Barnes, T. (2009). Experimental evaluation of an educational game for improved learning in introductory computing. *ACM SIGCSE Bulletin, 41*(1), 321–325.

Gunter, G. A., Kenny, R. F., & Vick, E. H. (2008). Taking educational games seriously: using the RETAIN model to design endogenous fantasy into standalone educational games. *Educational Technology Research and Development, 56*(5/6), 511–537.

Ho, P. C., Chung, S.-M., & Tsai, M.-H. (2006). A case study of game design for e-Learning. In Z. Pan et al. (Eds.), *Edutainment* (LNCS, Vol. 3942, pp. 453–462). Berlin Heidelberg: Springer.

Lahtinen, E., Ala-Mutka, K., & Jarvinen, H. (2005). A study of difficulties of novice programmers. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, June 27–29, 2005, Caparica, Portugal (pp. 14–18).

Law, K. M. Y., Lee, V. C. S., & Yu, Y. T. (2010). Learning motivation in e-Learning facilitated computer programming courses. *Computers & Education, 55*(1), 218–228. doi: http://dx.doi.org/10.1016/j.compedu.2010.01.007.

Lee, M.J., & Ko, A.J. (2011). Personifying programming tool feedback improves novice programmers' learning. *Conference on International Computing Education Research (ICER)* (pp. 109–116), Providence, RI, USA, August 8–9

Li, F.W.B., & Watson, C. (2011). Game-based concept visualization for learning programming. *Proceedings of the 3rd International ACM Workshop on Multimedia Technologies for Distance Learning* (pp. 37–42), Scottsdale, AZ, USA, December 01, 2011

Maragos, K., & Grigoriadou, M. (2011). Exploiting TALENT as a tool for teaching and learning. *The International Journal of Learning, 18*(1), 431–440.

Muratet, M., Torguet, P., Viallet, F., & Jessel, J.-P. (2011). Experimental feedback on Prog & Play: a serious game for programming practice. *Computer Graphics Forum, 30*(1), 61–73.

O'Kelly, J., & Gibson, P. (2006). RoboCode & problem-based learning: a non-prescriptive approach to teaching programming. *ACM SIGCSE Bulletin, 38*(3), 217–221.

Paliokas, I., Arapidis, C., & Mpimpitsos, M. (2011). PlayLOGO 3D: a 3D interactive video game for early programming education: let LOGO be a game. In *Proceedings of Third International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*, 4–6 May 2011 (pp. 24–31).

Phelps, A., Bierre, K., & Parks, D. (2003). MUPPETS: multi-user programming pedagogy for enhancing traditional study. *Proceeding of the 4th Conference on Information Technology Education* (pp. 100–105), Lafayette, IN, USA, October, 2003

Piteira, M., & Haddad, S. (2011). Innovate in your program computer class: an approach based on a serious game. OSDOC: Open Source and Design of Communication Workshop ACM, New York, NY, USA (pp. 49–54).

Salen, K., & Zimmerman, E. (2004). *Rules of play: game design fundamentals* (pp. 56–84). Cambridge: The MIT Press. pp. 304–350.

Yusoff, A., Crowder, R., Gilbert, L., & Wills, G. (2009), A conceptual framework for serious games. *The 9th IEEE International Conference on Advanced Learning Technologies* (pp. 21–23). July 15–17, 2009. doi: 10.1109/ICALT.2009.19.

Zualkernan, I. A. (2006). A framework and a methodology for developing authentic constructivist e-Learning environments. *Educational Technology & Society, 9*(2), 198–212.