ASTROCODE: A SERIOUS GAME FOR THE DEVELOPMENT OF COMPUTATIONAL THINKING SKILLS

Javier Bione[1], Pablo Miceli[1], Cecilia Sanz[2,3], Verónica Artola[2,4]

[1]School of Computer Science. UNLP (Argentina)
[2]Institute of Research in Computer Science III-LIDI. Scientific Research Agency of the Province of Buenos Aires (CICPBA), Argentina. School of Computer Science. UNLP (Argentina)
[3] Associate researcher CICPBA (Argentina)
[4] Doctoral Fellow CONICET (Argentina)

Topic Area: Educational Software & Serious Games- Educational/Serious Games

Abstract

Serious Games have been the focus of different research works in last years, as a consequence of their possibilities for educational scenarios. These games are used for purposes other than mere entertainment, but they are still motivating and attractive to users. More recent research works have also emphasized the potential of serious games for learning, skill acquisition, and attitude and behavior changes considering that they allow learners to experience situations that are impossible in the real world and provide engaging activities which are stimulating, generate strong emotions, require complex information processing, and provide challenges.

On the other hand, programming teaching is a growing area that is spreading beyond computer science courses of studies. Computational Thinking (CT) is related to a problem solving process that involves a number of characteristics and skills such as applying abstraction and algorithms design. CT is important in computer applications, but it can also be used to support problem solving across all disciplines. Today, there is a growing corpus of experiences whose goals include CT skills development.

This paper presents AstroCode, a serious game which is oriented to introduce players to CT skills such as algorithm design using variables and control structures. The description of the game and the test sessions held with different students of the National University of La Plata are detailed in this article. The first results show student engagement and motivation using these types of games in an educational scenario, as well as the difficulties they face in the use of control structures and how AstroCode helps students apply them.

Keywords: Serious Game, Computational thinking skills, 3D Games

1 INTRODUCTION

Games play a key role in the development and growth of individuals. This relation with games starts when a person is born, and is part of educational activities. For example, games allow safely experiencing situations that otherwise could not be experienced. Learning can be through experience, interaction, game design, and commitment to the game.

Since Abt [1] introduced the term “serious games” in 1970, different authors have tackled the concept. The authors in [2] mention that serious games are games with a goal other than entertainment. Several authors have related serious games to goals such as learning, training, attitude modification, developing certain competencies, and so forth [3, 4, 5].

The truth of the matter is, up to this day, there is no universally accepted definition of what a *serious game* is. Some of the definitions do not even characterize serious games based on developer intention when creating the game, but on how the individual plays the game; for instance, an individual could choose to play a rugby game in order to learn the rules of the sport [5]. In this paper, the definition proposed in [2] will be used, which describes serious games as those games whose primary purpose is not entertainment, enjoyment or fun. This does not mean that *serious games* are not entertaining or fun, it just means that they were created with other priority goal in mind.

Serious games can be categorized based on competence mastery within their goals, target audience, application area, or content formality level, be this content related to education, simulation or entertainment. Currently, there is a diversity of backgrounds in relation to serious games with educational goals or for the development of a specific competence. However, the focus of this article is on those serious games that are oriented to learning basic programming concepts, since a serious game developed to help students learn these concepts is presented. The game is called AstroCode, and it is a first-person game that allows building algorithms, using control structures and using data as part of the algorithms to build.

This article is organized as follows: Section 2 presents relevant background in relation to serious games oriented to programming teaching; Section 3 describes methodology aspects related to the work carried out, introduces AstroCode, its design and implementation, and describes the work sessions that were carried out to assess the possibilities of using AstroCode with students. Finally, in Section 4, the results obtained are discussed, and in Section 5, our conclusions and future lines of work are presented.

2 BACKGROUND

In recent years, the use of serious games has gained popularity in different educational institutions. On the other hand, there has also been a growing concern in relation to teaching and learning concepts and abilities related to computational thinking. Computational thinking is linked to the thought processes involved in problem and solution writing, where solutions are represented in such a way that they can be carried out effectively through an information processing agent [6]. There are several projects underway aimed at encouraging and promoting computational thinking. In [7], the authors present a study of various experiences carried out in Ibero-America involving the use of different technologies and programs. As part of this work, a number of games and experiences oriented to the learning of basic programming concepts, which helps develop computational thinking, were surveyed.

In [8], the game *Program your robot* is introduced. It is a serious game designed to help students practice basic programming concepts, such as building algorithms, debugging, and simulation. The goal of the game is building an algorithm-based solution that allows escaping a robot in different scenarios. The player can use iteration loops, conditionals and simple instructions, and can also create functions using all of these components for later reutilization. The algorithm is built by dragging and dropping blocks from a menu to the program. It should be noted that this paper also introduces an analysis of several serious games oriented to the development of CT skills. These games include: Robocode [9], for Java programming, and prog&play [10], a multiplayer real time strategy (RTS) game.

In [11], *Software Kids* is described, which is a serious game developed for Android devices, that allows introducing children mostly from ages 8 and up to basic object-oriented programming concepts such as classes, objects, attributes, methods, inheritance, polymorphism, abstraction, encapsulation, hiding, and modularity. Concepts related to software engineering were also considered, for example, common stages when solving programming problems, such as problem analysis, specification of requirements, design, implementation, testing, documentation, and maintenance. The game is based on answering questions or solving small statements, either by touching the correct answer or sorting elements on the screen. These questions and statements refer to programming concepts through everyday life elements.

RITA is also a serious game developed in Java and based on open code frameworks *OpenBlocks* and *Robocode*. Through the block programming features provided by *OpenBlocks*, *RITA* allows defining combat strategies for the virtual robot that fight on the battlefield of *Robocode*. *RITA*'s grammar is based on a collection of graphic blocks. Students build their programs by dragging and dropping blocks and interlocking them. Syntax does not include punctuation marks, or English words, which are commonly used in programming languages. The academic prerequisites to play *RITA* are minimal, so any high-school level student can play it without much trouble [12]. Finally, EPIT [13] and EPRA [14] were reviewed, two games oriented to the teaching of basic programming concepts that involve using current human-computer interaction paradigms, such as tangible interaction in the case of EPIT, and augmented reality in the case of EPRA. These games are attractive to students and have yielded good results in terms of motivation to learn, among other aspects.

3 METHODOLOGY AND DESCRIPTION OF ASTROCODE

This article is part of a relational applied research process that includes digital technologies, in particular the use of games, for teaching and learning processes. More specifically, this work was carried out as part of a graduate dissertation of the School of Computer Science, National University of La Plata, Argentina.

The methodological process used in this research consisted of:

- a review of bibliography on serious games, which resulted in an analysis of definition issues in our Introduction;
- a review of the background of serious games used to teach programming concepts, as well as an analysis of the possibilities offered by these games; this information is discussed in our Background section, with some details about some of the games that were reviewed;
- definition of the requirements of the serious game to be designed based on success aspects of other games with a similar goal, but also considering additional aspects that were not taken into account by these other games.
- serious game implementation based on previously established requirements; the implementation has already been carried out and there is currently a second version available, which is discussed in Section 3.2;
- planning test sessions for the game with students in a specific educational scenario, considering usability, possibilities for the acquisition of CT skills, and intrinsic motivation during use as assessment lines. Section 3.3 includes a detailed description of how these sessions were planned and carried out.
- analysis of the results obtained during these sessions and planning a proposal for improvement. The most relevant results are discussed in the corresponding section.

3.1 Requirement Definition Before Designing AstroCode

After reviewing both bibliography and serious games oriented to programming teaching, some common features were found that seem to be successful when it comes to developing CT skills, and other aspects were identified that these previous projects did not take into account and that we believe could be motivating and promising for young students. Below, there is a list of some of the features found that were considered of interest to add to the development of the serious game proposed:

1. Creation of algorithm by dragging and dropping blocks that represent instructions. This feature was used in [8] and [10]. It is also included in visual programming environments, such as *Scratch*, which is widely used in the international educational scene.
2. Use of game levels that add a progressive level of complexity in terms of competence acquisition in the development of algorithms.
3. Algorithm compilation and error detection, as explained in [8].
4. Awards and rewards that favor student motivation as gamification technique.

There is also a number of additional requirements to include as motivational and/or learning elements:

1. Player versus player (PvP) competition to compete for a place in the rankings of players shown in the game's website.
2. Possibility for educators to configure levels or scenarios within the game to customize it for a given educational context.
3. Assigning categories to participants within the game, based on the style and quality of the algorithm built (considering the use of appropriate control structures and the least possible number of steps for a given run within a game scenario).
4. Use of first person 3D scenarios to increase participant engagement in game dynamics.
5. Addition of decision-making levels before building an algorithm, related to the selection of a given robot that is appropriate for the scenario at hand (the instructions supported by each robot should be analyzed to choose the one that corresponds to the solution).

These are just a few of the requirements proposed before starting the design and development stages for AstroCode. They are described in the following section.

3.2 Description of AstroCode and Its Implementation

3.2.1 Introduction to AstroCode

AstroCode presents the story of a spaceman exploring different planets. The spaceman receives a mysterious signal and he has to travel to another galaxy to investigate it. On the way to his destination, a malfunction in the ship forces him to land on an unknown planet. Before impacting the planet, the astronaut is ejected from his ship in a rescue capsule. Now, he must go through different scenarios in this planet, collecting pieces of the ship, and recovering them to complete his mission. The astronaut has a number of robots that will help him gather the pieces (this means "solving the scenario"). The game proposes that, for each scenario, one robot has to be selected according to its capabilities to solve it. The selection of the robot presents a first challenge for the player, who must have the skills to evaluate the instructions each robot understands (its language) and if it can be used to solve that particular scenario. To recover the pieces, the player must give a sequence of instructions to the selected robot, that is, build an algorithm to make the robot move and overcome any obstacles found in its way. Scenarios have a gradual level of complexity, and the player will become increasingly involved with algorithms that are also increasingly harder to solve.

Game play can be toggled between free first person movement on the different planets and scenarios where the story develops, and programming scenarios, where the players must build an algorithm that allows a given robot solve the challenge. These algorithms are built by dragging and dropping blocks that represent instructions, conditions and control structures. Some space characters have also been introduced as story companions and to increase game interaction. For instance, an alien that asks for help to learn some aspects linked to the history of Computer Science.

Figure 1 shows the programming scenarios that create learning instances for players and will test their knowledge by asking them to build an algorithm that allows a given robot to get all pieces in the map. Throughout the game, different programming scenarios are found, each posing a different level of difficulty. Upon starting each scenario, players will be presented with a number of hints introducing the concepts and difficulties for that scenario, accompanied by video-tutorials that help understand the various aspects of the game.

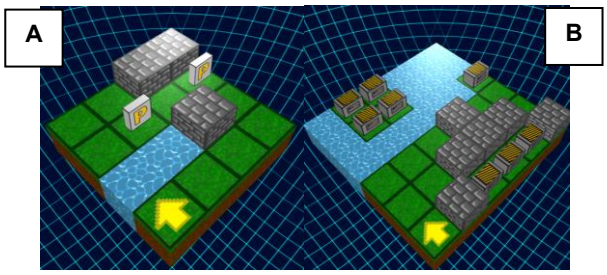

Figure 1. A shows a low-complexity scenario with various types of blocks (water blocks, stone blocks) and the two pieces (indicated with a P) to be retrieved. B shows a more complex scenario, with the addition of chests that could be hiding pieces

To solve each scenario, there is a number of robots, which are listed in Table 1. Initially, not all robots are available. They are progressively unlocked as scenarios are solved.

Table 1. Currently available robots in AstroCode to solve scenarios.

Robot	Abilities
Explorer	Explorer is the robot that the player meets upon starting the game and, as such, it is available at start. It will only understand basic movement instructions, namely: *move forward*, the robot moves forward one square in the direction it is facing; *right*, the robot turns clockwise on its own axis; *left*, the robot turns counterclockwise on its own axis; *grab*, the robot tries to grab a piece in the square located directly in front of it.
Laserbot	In addition to basic movement instructions, it understands instructions that allow destroying the blocks of any walls obstructing its way. Specifically, this robot adds the command *destroy*, which, as its name denotes, allows destroying a wall block located right in front of

	the robot.
Drone	The Drone robot can fly, which is useful to go around obstacles with ease, since it can fly over obstacles that will not let other robots through. In addition to basic movement instructions, this robot can open chest blocks simply by flying over them. This robot also understands the condition *PiecePresent*, which can be used within an *If or While* control structure to determine if the square directly below the Drone has a piece in it or not
Dogbot	The Dogbot understands the instruction *dig*, which allows the robot to remove dirt from a "dirt" type block. Dirt blocks have a random volume of dirt, which means that the robot will likely have to use the *dig* instruction more than once. To resolve this problem, the robot also understands the condition *DirtPresent*, which determines if the square below the robot has dirt or not. This condition can be used together with the control structure *While* to dig all the dirt present in the square.

Figure 3 shows an example of the interface, where the player solves a programming scenario by dragging instruction blocks from the selected robot. Upon successful resolution, feedback on the quality of the algorithm built is provided, which is presented as a medal. If the player does not obtain the best medal, that is, if the algorithm is not the best possible one, the medal obtained is accompanied by tips on how to improve solution quality. The medal awarded is calculated based on several factors, including the number of instruction blocks used, the number of instructions executed by the robot, the use of control structures, whether all squares with chests were checked, and the number of warnings shown in the console. The medals available for each scenario are, from lower to higher merit: *beginner programmer medal*, *intermediate programmer medal*, *advanced programmer medal, and intergalactic programmer medal*.

Figure 2. On the left, the instruction blocks available for the selected robot, which is on the upper left, are shown. Blocks are dragged to the program section, and program controls are at the bottom. On the right, a scenario to be solved with the feedback area below it is shown.

To develop the game, Unity 3D was selected and C# programming language, and, therefore, the Object-Oriented Programming paradigm was used. To create the 3D models that are used in the game, 3D Blender was used as modeling tool, since it is free and has a large developer community and examples to get information.

The game is available in its own website (http://www.astrocodigo.com/) for anyone to download. It can be run on any computer with Windows, Linux or MacOSX operating system.

On the other hand, an editor has been developed that can be used by teachers to generate the scenarios of the game and give the complexity that is required for the work with its students. Next, the scenario generator feature is described.

3.2.2 Scenario Generator

The introduction of a scenario generator to customize scenarios is based on the need of offering various possibilities for the educational context, so that scenarios can be customized for different situations and educational levels. Thus, a tool was conceived that would allow the design and development of new programming scenarios, which would in turn allow defining new goals and challenges that are not present in the original game.

The generator is a web tool that can be accessed from any computer with a browser and Internet access. This tool works together with an online database that stores the scenarios created with the generator, which are then retrieved by the game to be shown in the customized scenarios section of

the start menu. Figure 3 shows the generator interface, with the different types of blocks that can be added.

Figure 3. This figure shows the generator, in which the size of the scenario and the types of blocks to add can be selected, as well as the number of pieces that have to be retrieved.

3.3 Testing

Test sessions were first carried out with the participation of volunteer users who, during game sessions and using the thinking aloud technique, contributed ideas and enhancements for the game. These sessions took place during a period of 3 months after the first executable version of the game was available. As a result of this process, for instance, help tutorials, which initially included only text and images, became tutorial videos. Some buttons were also changed, and several aspects of the interface required attention to address some weaknesses detected by testers.

Once a stable and debugged version of the game was produced, a test session with students was organized. In the context of pre-entry course to the School of Computer Science of the UNLP, which allow students become familiar with the main topics of the career, students were invited to participate in a test session. Initially, students from the course of studies in Computer Science were enrolled, but later on, an Electronic Engineering student and two Physical Education students were also invited to collect feedback from players who had no relation to Computer Science. There was a total of 10 participants who were mostly between 17-20 years old and male.

The session was planned with the following stages: a. introduction to the game and its development context, b. solving 6 scenarios in the first planet of the game, c. gathering data through a survey and interviews. The session was videotaped and surveys were carried out online.

The variables that were considered during the session were as follows: difficulties during the game as reported by the students, time required to solve each scenario, use of control structures, selection of the appropriate robots, and medals obtained. On the other hand, surveys were focused on analyzing student motivation through an IMI (Intrinsic Motivation Inventory) questionnaire [15] as well as considering usability and user satisfaction aspects. During the interviews, a more informal approach was taken to gather information about what the students thought of the game, and identify any issues that might have been overlooked in the other information-gathering instruments.

4 RESULTS

The survey showed that 80% of the participants were familiar with the programming concepts presented in the game. The remaining 20% answered that they had no previous knowledge of the concepts presented. It was observed that 8 of the 10 participants solved all scenarios, and that the 2 remaining participants were able to solve 4 and 5 out of the 6 scenarios, respectively. As regards the perceived ease of use of the game, participants rated the game by assigning a score on a scale between 1 and 5, where 1 was very easy and 5, very hard. Since the purpose of the game is introducing programming concepts to young students before they have had any contact with computer science, these results were expected. Forty percent of survey responders assigned a score of 2 to the game, which would translate as “easy,” while the remaining 60% assigned a score of 3, which would translate as “medium difficulty” (see Figure 4). Therefore, it can be concluded that for students in the same age range as our participants, the game poses a moderate difficulty.

The survey also asked participants to rate game educational perceived usefulness using a similar scale as the one in previous questions. Figure 4 shows that 90% of the participants answered 5, meaning that they found the game to be “very useful,” and only 10% answered 4, which translates as “quite useful.”

Based on how participants answered this question, it can be concluded that the game is a very useful educational tool. Similarly, during the test session, it was observed that those students with less knowledge asked how to repeat instructions, which pointed to the need of using a control structure, even if they were not familiar with this concept. They also asked how to input conditionals, when they realized they needed to use them. This was an interesting aspect of the game, since students were able to identify the need of using these instructions, while those who were already familiar with the concepts used them of their own accord.

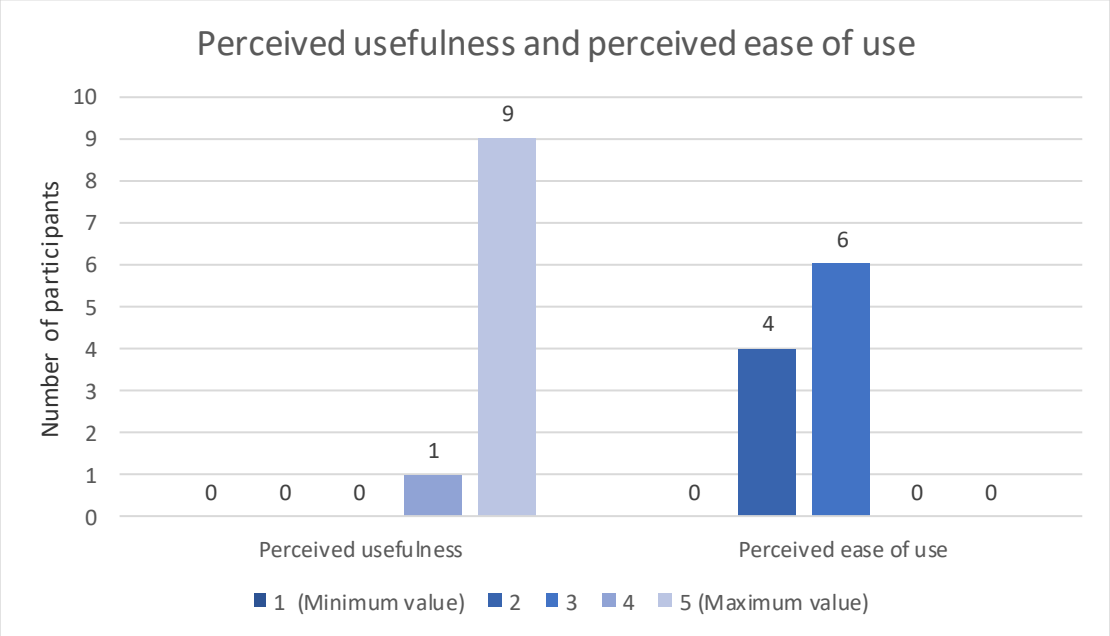

Figure 4. Perceived game usefulness from an educational perspective versus perceived easy of use.

Participants were also asked how they thought the game motivated young students to choose computer science related courses. Using a 1-5 scale similar to the previous ones, 1 being “not motivating at all” and 5 being “very motivating,” students were asked to rate the game once again. In this case, 40% of the participants answered with 5, indicating that they found the game could be “very motivating” for young students to choose computer science related courses. Fort percent answered 4, which would translate as “quite motivating,” while the remaining 20% answered 3, meaning they found the game “somewhat motivating.”

The next question asked participants to identify which of the concepts the game was trying to convey they felt they had learned or reinforced. The options provided were as follows:

- Building programs. (8 people chose this)
- Decision (control structure IF). (8 people chose this)
- Repetition (control structure REPEAT). (7 people chose this)
- While (control structure WHILE). (8 people chose this)
- Hardware and Software. (2 people chose this)
- Programming languages. (3 people chose this)
- Other. No one chose this

It was observed that some students tried to use a decision control structure instead of using a "while" control structure. These problems were discussed during the interview phase and students felt that they understand better the use of control structures.
Robot selection, even if participants did have doubts at first, was a motivating element, and it allowed students to approach the concept of language.

With the data obtained from the IMI questionnaire, it was determined that the level of interest/enjoyment was very high, with healthy competition among participants and a very high general feeling of being able to choose. Therefore, it can be concluded that the motivational aspect of the game is satisfactory. The pressure/tension index, representing the negative factor in the sub-scales, was low to medium.

5 CONCLUSIONS AND FUTURE WORKS

AstroCode is considered to be a contribution to the community, in a time in which the development of CT skills has gained popularity among those who analyze and participate in current and future education. Game dynamics generated interest and motivated young students. It was found that it favors an intuitive approach to the concept of control structures and the need to build algorithms. It also allowed discussing the concept of algorithm and language with the students. Participants indicated that the game helped them better understand decision, repetition and iteration control structures. Also, all participants said they would recommend the game to their classmates and that they would like to keep playing it. They were interested in knowing how the story progressed, which indicates that the story is in itself a successful motivation element. On the other hand, the fact that the pressure/tension index got to half scale opens the door to a further analysis of session setup to determine which aspects made participants feel this way. Additional testing sessions will be held and the game will be further promoted to carry out a more detailed analysis and obtain more conclusive results. The scenario generator will also be tested with educators and interested members of the general audience. Thus, AstroCode is expected to continue to evolve.

ACKNOWLEDGEMENTS

Special thanks to the REFORTICCA (Resources for Empowering ICT, Science, and Environment Educators) project, funded by the Scientific Research Agency of the Province of Buenos Aires (CICPBA), which will allow promoting AstroCode among secondary level educators and students in the Province of Buenos Aires, Argentina.

REFERENCES

[1] C. Abt, *Serious games*, Viking Press: 1970.

[2] D. Michael and S. Chen, *Serious games: Games that educate, train, and inform*. Thomson Course Technology: 2005.

[3] E. Boyle, T.M. Connoly and T. Hainey: “The role of psychology in understanding the impact of computer games”, in *Entertainment Computing*,Vol. 2, pp. 69-74, 2011.

[4] B. Sawyer, B. and P. Smith: Keynote Address. “The Second European Conferences on Games-Based Learning”. UOC, Barcelona, Spain, 2008.

[5] R. Dörner, W. Effelsberg, S. Göbel and J. Wiemeyer: *Serious Games. Foundations, concepts and practice*. Springer International Publishing. Germany: 2016.

[6] J.M. Wing: Computational thinking. In *Communications of the ACM*, 49(3), 33–35, 2012. Retrieved from:http://research.microsoft.com/en-us/um/redmond/events/asiafacsum2012/day1/Jeannette_Wing.pdf

[7] M. Sarmiento, G. Gorga and C. Sanz. “Análisis de experiencias y estrategias educativas con TIC para el desarrollo del pensamiento computacional en estudiantes de secundaria y primeros años de universidad en Iberoamérica”. *Trabajo Final de Especialización en Tecnología Informática Aplicada en Educación*. 2017.

[8] L. Bacon, C. Kazimoglu, M. Kiernan and L. Mackinnon L.: *A serious game for developing computational thinking and learning introductory computer programming*. University of Greenwich. London, United Kingdom: 2012

[9] Robocode: *Open source educational game*. Retrieved from: http://robocode.sourceforge.net.

[10] M. Muratet, P. Torguet, F. Viallet and J.P. Jessel: “Experimental Feedback on Prog&Play: A Serious Game for Programming Practice”, in Computer Graphics Forum, 30(1), pp. 61-73, 2011.

[11] M. Leon-Sigg, S. Ramírez-Rosales, S. Vazquez-Reyes and J.L. Villa-Cisneros: *A serious game to promote object oriented programming and software engineering basic concepts learning*. Universidad Autónoma de Zacatecas. Mexico: 2016.

[12] M. Brown Bartneche M., L. Fava, S. Gómez, I. Miyuki Kimura and C. Queiruga: *El juego como estrategia didáctica para cercar la programación a la escuela secundaria*. Laboratorio de Investigación en Nuevas Tecnologías Informáticas, Facultad de Informática, Universidad Nacional de La Plata. Argentina, 2014.

[13] V. Artola, C. Sanz, G. Gorga, P. Pesado: *"*Diseño de un juego basado en interacción tangible para la enseñanza de programación"*, in Congreso Argentino de Ciencias de la Computación: 2014*.

[14] N. Salazar Mesia, C. Sanz and G. Gorga: "Augmented Reality for Programming Teaching. Student Satisfaction Analysis", in *Proceedings of the 2016 International Conference on Collaboration Technologies and Systems*, pp. 165-171. Orlando, Florida, USA, 2016 – ISBN: 978-1-5090-2300-4/16,2016 IEEE DOI 10.1109/CTS.2016.43

[15] *IMI* (*Intrinsic Motivation Inventory*). Retrieved from: http://www.selfdeterminationtheory.org/