A Review of Games Designed to Improve Introductory Computer Programming Competencies

Adilson Vahldick, António José Mendes, Maria José Marcelino
Centre for Informatics and Systems – University of Coimbra
Coimbra, Portugal
adilson.vahldick@udesc.br, toze@dei.uc.pt, zemar@dei.uc.pt

***Abstract*—Learning computer programming is not simple for many students. They have to develop several complex skills to be able to understand programs and, more important, to create programs that solve problems. This means it is important that students have a high motivation level, so that they engage in that work and do not get frustrated with the natural errors they will make in this process. Digital games are often used in educational contexts to attract and retain students. In literature and on the web, we can find many games related strategies that aim to support learning in introductory computer programming courses. One of these strategies is the using games approach: asking students to play games that include problems that must be solved in order to progress. This paper presents a list of 40 games classified by type and highlights the skills and topics supported by them. We hope this work helps teachers to choose games as part of their teaching strategies, as alternative or complementary exercises to their students.**

Keywords—games; introductory programming; novice; teaching;

I. INTRODUCTION

The global demand for software developers is rising, but the number of computer course's enrolments is decreasing [1]. To make matters worse, the failure and dropout rates are high, mainly in introductory computer programming courses [2].

One reason for this is the high amount of practice required to learn to program and the need for an high students' engagement on extra classroom activities [2]. However, often they have a lack of motivation to engage in these tasks. Games have been used as motivational strategies in introductory programming teaching, as they may allow interactive entertained and creative experiences [3]. Two approaches have been used: creating and playing games. In the first approach students are asked to develop small games in order to apply the programming concepts [4, 5]. In the second approach the students play games to reinforce and practice concepts and programming skills [1, 6, 7]. The main idea is to motivate students to the learning activities, shortening the time between theory and practice, and bringing together abstract concepts and concretes activities. In this paper, we have concentrated on the use of existing games for computer programming education purposes. Thus, game authoring tools and experiences where students create games were not considered in this work.

There are several published works and available games on the web aiming to support introductory computer programming learning. With the purpose to foster the use of games, raise teachers' awareness about this possibility, and know better what is available, we made a review that identified 40 games related to the introductory programming course's subjects and competencies.

In section II we discuss the skills that students need to develop to learn to program. Section III briefly introduces the producing games approach and a classification of game types that may be used in programming learning activities. The research questions and the methodology used in this literature review are described in section IV. The games are reported, sorted and discussed in sections V, VI and VII.

II. COMPETENCIES, SKILLS AND CURRICULUM

Before defining the curriculum and the learning objectives, it is required to identify the students' expected competencies. There are three essential competencies for the novice programming students: comprehension, writing and debugging [8–10]. These competencies are transversal, as the students do not appropriate the whole competence at a time. Each competence is formed by a set of skills, which are the actions that a student will learn for mastering a competence. The skills may overlap among the competencies. The subjects are defined to guide the pedagogical practices and to support activities for providing the skills. Curriculum is the set of competencies, skills and subjects.

To have a functional comprehension of a program the student should know how the computer behaves in relation to a program. The biggest weakness of the novices is the abstraction capacity about how the machine interprets a program [8, 11]. For instance, the lines of code run one at a time, and the computer does not have the ability to understand commands that are not explicitly coded. When the students get used to the notional machine, then they can study a programming language syntax and semantic to start learning how to program [12–14]. At the same time, students should be taught graphical models and representations to help program visualization, making easier to understand the relation between code parts [9, 12]. As experience is gained after using the code lines and graphical representations, the strongly related parts give meanings to the students and they perceive solution patterns that can be applied in some kinds of problems. The graphical representations can strengthen memorization of those structures and their applications [12].

978-1-4799-3922-0/14/$31.00 ©2014 IEEE

Code debugging includes skills as program test, tracing, bug finding and fixing. There is a stronger relationship between code comprehension and debugging than between code writing and debugging [12, 15, 16]. The students are more confident in writing when they have better knowledge and debugging skills [17].

To write computer programs novice students need to apply knowledge in a new situation. They have to transform mental solutions into computer programs [12]. The student task is not just to write simple programs, but also to solve problems by coding. The student needs to develop essential skills as abstraction [18], generalization [8], decomposition [19] and problem solving strategies [9]. To write code the student should understand the problem and abstract a model, divide the problem into smaller parts, decide a strategy to solve each part, and apply or adapt already known solutions [2, 20]. In the literature this competence is often mentioned as the source of most students' problems [8, 9, 11]. Many times the students show language syntax and semantic knowledge, but they have problems on applying it to solve problems [20].

ACM and IEEE included a new area in Computer Science Curricula 2013 guide: Software Development Fundamentals (SDF) [21]. This new area was spread over other knowledge areas in the previous version. The units *Fundamental Programming Concepts*, *Fundamental Data Structures*, *Algorithms and Design*, and *Development Methods* compose SDF. The topics of the two first units, usually included in CS1, are shown in Table I. The other two units have topics normally included in the next courses [22]. For instance, unit *Fundamental Data Structures* includes *Abstract data types*, *References and aliasing* and *Linked lists*. The unit *Algorithms and Projects* has subjects as *Problem-solving strategies*, *Abstraction* and *Program decomposition* characterized as skills subjects, due this are also not included.

TABLE I. SOFTWARE DEVELOPMENT FUNDAMENTALS

Units	Topics
1. Fundamental Programming Concepts	1.1. Basic syntax and semantics of a higher-level language 1.2. Variables and primitive data types 1.3. Expressions and assignments 1.4. Simple I/O including file I/O 1.5. Conditional and iterative control structures 1.6. Functions and parameter passing 1.7. Recursion concept
2. Fundamental Data Structures	2.1. Arrays 2.2. Records/structs 2.3. Strings and string processing

III. GAME-BASED LEARNING IN INTRODUCTORY COMPUTER PROGRAMMING COURSES

There are two approaches to use games to support programming learning: creating games and playing games. The second approach will be discussed in sections V and VI.

In the first approach, students are asked to design and develop small games. For example, the student may have to create a game where a player controls a robot to collect coins during a limited amount time. Games are created using particular environments and authoring languages (for example, Alice[1] and Scratch[2]), or using conventional programming languages (as Java in Greenfoot[3], JKarel[4], Furbot[5]). There are also *microworlds* where students have to program some behaviour (RoboMind[6], Scalatron[7] and PlayLogo 3D [23]), and others where the students program robots that compete in a battle (Robocode[8]).

Wassila and Tahar compiled game types according to the skills to develop in the students [24]. With this information, the teacher can decide the applicability of the game into his/her classroom time and their relation with the class goals. There are three big categories of games: action games (sports, combat, platform, mazes and interactive movies), strategy games (adventure, resource management and war), and hybrid games (simulations, real-time adventure and strategy games). The most suitable game types for problem resolution are puzzles, simulation games, strategy games, adventure games, artificial life and management simulations [3, 24].

IV. RESEARCH QUESTIONS AND METHODOLOGY

There are several works published about games developed to support computer programming learning. However, we did not find papers that organize these type of games according to the skills they are supposed to promote. We undertook a systematic literature review following the principles presented in Brereton et al. [25]. The research questions addressed by this work are:

1) How many and which are the games available to assist introductory programming learning published in literature or on the web after 2000?

2) Which are the abilities and topics covered by these games?

3) What features are missing to better support introductory programming learning?

To answer these questions we need a sample of introductory programming games. We searched them in digital libraries through ACM Digital Library, Science Direct and IEEE Xplore using the terms *game*, *introductory programming*, *computer programming* and *novice programming*. Next, we scanned the results and considered the papers that describe games used, proposed or developed to support introductory programming learning. The same was made with the papers referred by them, perhaps not indexed in the libraries above. We considered also three Brazilian informatics conferences: Brazilian Computer Society Congress, Brazilian Informatics in Education Conference and Brazilian Digital Entertainment and Games Symposium.

[1] http://www.alice.org
[2] http://scratch.mit.edu
[3] http://www.greenfoot.org
[4] http://www.cs.tufts.edu/comp/10F/JKarel.htm
[5] http://furbot.sourceforge.net/
[6] http://www.robomind.net
[7] http://scalatron.github.io
[8] http://robocode.sourceforge.net

Afterwards, we searched the terms *game* and *programming* in commercial databases (as AppStore and Google Play), *programming* in sites with Flash games and on the web. We found also games described by their developers in GamaSutra[9]. We have gathered only the games that explicitly relate to computer programming learning. Several puzzle games may improve reasoning, but they were not the focus of this work and were not added to the list of the games (Table II).

For each game, we: a) read the paper, when the game was described in a journal or conference proceeding; b) read the webpage, when the game was available for download and there was enough information identifying it; or c) we played it when the information was not enough.

For this work, we considered games where the challenges, or the missions, are embedded in the game and it can check the player answers. This means that the student can be autonomous and play (and learn) without the teacher's intervention. However, there are other games described in some papers, which need teacher intervention to check answers. These games were not considered in our study, as they do not support students' autonomous work.

Based on the sample, we classified the games according the competences and topics discussed in section II and the types discussed in section III.

V. RESULTS

The most popular type found was action games where the player programs the movements of a robot, a turtle or other kind of character in some simulated world. The missions are to reach positions in the world, or to collect some quantity of objects. The movements are programmed through a simplified programming language, as the LOGO language from Seymour Papert. In all the games in this type, the programs are created by drag and drop commands from a toolbar. This eliminates syntax errors. Although there are some works directed to undergraduate students, the target audience of almost all these games are kids. In this paper we are proposing the *LOGO-like* type due the high quantity of this kind of game. The problem-solving skills prevail in this type, promoting algorithm and logic reasoning [26]. We found 19 games of this type.

The second most popular type is *Adventure Games*. The player commands a hero to explore world, to collect objects and to interact with other characters controlled by the game [27]. We found 13 games in this type.

The remainder games found belong to several types, such as simulations, real time strategies and maze games. For this reason, we grouped them in only one type called *General Puzzles*. We found 8 games in this type.

Fig. 1 shows the type distribution of the 40 games found in this research and Table II lists them indicating some additional information:

- Type of the game: "L" for *LOGO-Like*, "A" for *Adventure* and "G" for *General*;

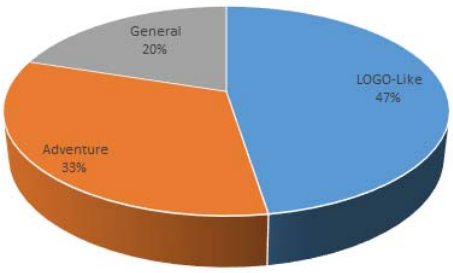

Fig. 1. Introductory programming games by type

- The platform where they run, Windows, iOS (tablets and smartphones), Web, Android and Linux. One of the games is indicated by "?" because we did not identified;
- The three competences (C-comprehension, W-writing and D-debugging) according the activities and tasks;
- Some of them cover specific subjects. The column topics is filled as stated by the Table I;
- The programming language applied to write the solutions in these games.

About the competence column, the games in the comprehension competency were established by activities like match, fill-blank and choice one instruction to complete some code. In the writing competency, we included the games with activities such as:

- Solve traditional assignments (as mean calculation and sort elements);
- Program the game elements' behaviour through textual or visual commands;
- Drag & drop code blocks to write a program.

In the debugging competency, we included games that have tasks to track and assess changes in variable values. Moreover, some games have tasks to fix lines of code to make the program work. In addition, other games have tasks to change parameters of conditional or repetition structures.

Some of the games are designed to meet particular subjects. We mapped them in the column topic. This classification was also made based on the games' activities:

- 1.2. Variables and primitives data types: variable declaration (match the declaration, name and value), variable type (match the value and type) and variable initialization;
- 1.3. Expressions and assignments: variable values assignment and manipulation;
- 1.4. Simple I/O including file I/O: write code to store and retrieve heroes items;
- 1.5. Conditional and iterative control structures: reorder a block of code with loops and nested loops;

[9] http://gamasutra.com/blogs/RobLockhart/20130905/199667/Games_That_Teach_Programming_A_Brief_Overview.php

TABLE II. LIST OF GAMES

Game	Type	Platform	Comp.	Topics	Language
APIN [29]	L	Win	W	1.5	Textual Block Graphics
BootLogic http://botlogic.us/play	L	Web	W	-	Visual Block Graphics
Baralho das Variáveis [30]	G	Web	C	1.2, 1.3, 2.3	-
Cargo-Bot http://twolivesleft.com/CargoBot	L	iOS	W	1.6, 1.7	Visual Block Graphics
Catos Hike http://hwahba.com/catoshike	L	iOS	W	1.5	Visual Block Graphics
Coddy Luck http://appike.com/coddy	L	iOS	W	1.5, 1.6, 1.7	Visual Block Graphics
Code.org http://learn.code.org	L	Web	W	1.5, 1.6, 1.7	Textual Block Graphics
Code Combat http://codecombat.com/	A	Web	W	-	JavaScript
Code Spells https://sites.google.com/a/eng.ucsd.edu/codespells/	A	Win	W	-	Java
Colobot http://www.ceebot.com/colobot/index-e.php	A	Win	W	-	Proprietary inspired in C++, C# and Java
CMX [31]	A	Win	W	-	Java
Cube Game [32]	L	Web	W	1.5	Visual Block Graphics
Daisy the Dino http://www.daisythedinosaur.com	L	iOS	W	1.5	Textual Block Graphics
Dream Coders [33]	A	Java	W	1.4, 1.5, 2.1	Java
EleMental: The Recurrence [34]	G	Win	W	1.7	C#
Entrando pelo Cano [35]	G	Web	C	1.2	-
Gidget [7]	L	Web	W	1.5	Textual Block Graphics
IA Game [36]	A	iOS, Win	D	-	-
Kodable http://www.surfscore.com	L	iOS	W	1.5, 1.6, 1.7	Visual Block Graphics
Light-Bot 2 http://armorgames.com/play/6061/light-bot-20	L	Web	W	1.5, 1.6, 1.7	Visual Block Graphics
Machinist-Fabrique http://learntocode.biz/	G	Win	W	-	Textual Block Graphics
Move the Turtle http://www.geekkids.me	L	iOS	W	1.5, 1.6, 1.7	Visual Block Graphics
Prog & Play [6]	G	Win, Linux	W	1.5	C, C++, Java, Ocaml, Ada and Compalgo
ProGame [37]	A	Win	CD	-	-
Program your robot [26] http://www.programyourrobot.com	L	Web	W	1.5, 1.6, 1.7	Visual Block Graphics
Project Orion [38]	A	Win	W		Configurable by the teacher
RoboCom http://www.cubesteam.com/Game-RoboComBasic.html	L	Win, iOS, Android, Linux	W	1.5, 1.6, 1.7	Visual Block Graphics
RoboLogic http://www.digitalsirup.com/apps/app_robologic.html	L	iOS	W	1.6, 1.7	Visual Block Graphics
Robotimov [39]	A	Win	CD	-	-
Robozzle http://robozzle.com	L	Web, iOS, Android	W	1.5, 1.6, 1.7	Visual Block Graphics
Ruby Warrior https://www.bloc.io/ruby-warrior	G	Web	W	-	Ruby
Saving Sera [1]	A	Win	WD	1.5	Proprietary language
Space Goats [40]	G	iOS, Android	W	-	Visual through blocks
Takkou [41]	L	?	W	-	Textual Block Graphics
TALENT [42]	L	Web	W	-	Textual Block Graphics
The Catacombs [1]	A	Win	C	1.5	-
Train B&P [43]	G	Win	W	-	Proprietary language
Tynker Lost in Space http://www.brainpop.com/games/tynkerlostinspace	L	Web	W	1.5	Visual Block Graphics
World of Variables [44] http://hrast.pef.uni-lj.si/~svet_spremenljivk/	G	Win	C	1.2	-
Wu's Castle [45]	A	Win	D	2.1	-

- 1.7. Recursion concept: complete a code to transverse a tree structure;
- 2.1. Arrays: transverse an array changing the loop parameters;
- 2.2. Strings and string processing: operations with strings to achieve a word gave by the game.

The programming language is important to help teachers to select the game according the language adopted in their courses. We divided the *LOGO-Like's* block graphic commands in two groups: the visual commands use icons and images, and in the textual the names of the commands are explicitly showed.

VI. Discussion

In practical terms, the teachers' first consideration is the availability of the games. Some games are described in published papers, but accessing them depends on contacts with the authors. Those papers generally describe the game, how it was used and the results obtained. In contrast, there are games available on the web for download and use. In this case, the teacher is responsible to conduct the experiment without any support. We found only one game in both, available on the web and described into a published paper: *Program your robot.*

LOGO-like games have nice interfaces in the iOS platform. However, the best choice is web games, because they are less restricted due to the electronic gadgets (smartphones and tablets) capabilities. Four games stand out in web platform: *LightBot 2*, *Robozzle*, *Program your robot* and *Code.org*. The first two have similar functionalities, including the way in which students should solve problems. Neither of them has iterative structures, proposing the use of recursive calls instead. Moreover, there is an editor in both games, allowing teachers to insert more problems. *Robozzle* has iOS and Android versions. The way to solve the problems in *Program your robot* follows the programming language approach, because conditional and iterative blocks need to be created. *Code.org* is a recent effort in partnership with United States government to promote computer education among the children. This game covers a wide range of topics, from step by step line execution to functions and parameters. The player uses blocks' drag & drop to write programs, in a similar way to Scratch.

Any person can play *LOGO-like* games, from children to adults, even if they are not in the context of an introductory programming course. However, the other two types are more specific to introductory computer programming courses.

The games focused on the comprehension competency can be applied as learning reinforcement or concepts visualization. The debugging games have essentially bug fix and change parameters tasks.

Most code writing games are available on the web. These games can be adopted as an alternative to traditional assignments. They provide missions and the player must code the hero (or other characters in game) behaviour to achieve the mission. The player progress is automatic monitored by the game, without teacher intervention. The student solves the problems using programming motivated and engaged by the game.

Teachers usually prefer to select a tool that supports the same programming language adopted in the course. Some games go a step further in this aspect, as they support more than one language. For example, the game Project Orion has an external tool, which allows the teacher to configure the language to be used. Prog & Play allows selection among four traditional programming languages and two scripting options. The games Code Combat, Code Spells, Colobot, CMX, Dream Coders and EleMental adopt just one language, as C++, C# or Java. Machinist-Fabrique has rich challenges, and uses a block-based graphical language similar to Scratch. We do not classified it as *LOGO-Like* because of its activities. The puzzles are not based in move a character from a place to another, but controlling machines as winches, claws and flip floppers.

The development of the debugging competence is natural in code writing games, because the character behaviour makes the same role as the variables in traditional programs. These games expose the state of the variables through animation of characters' actions.

VII. Conclusions

In this paper were evaluated some games published in the last decade, which claim to support the development of competencies in introductory computer programming learning. The research was based in the literature and on web available games. We wanted to know more about what is available. Also we believe the collected information may be useful to teachers that are considering to use a game-based approach in their courses.

Now we try to answer each of the research questions put in the beginning.

1) How many and which are the games available to assist introductory programming learning published in literature or on the web after 2000?

The answer is in the section V. We found 40 games and classified them in three types: *LOGO-like*, *Adventure games* and *General puzzles*. The list of *LOGO-like* games is also in the section V.

2) Which are the abilities and topics covered by these games?

We answered this question in sections V and VI. The *LOGO-like* type is more adequate to algorithmic reasoning development. The other two types are distributed in the support to the three competencies identified.

Most the games (47%) belong to the *LOGO-like* type, and we can observe a trend in development to mobile devices, mainly to iOS platform.

The main goal of an introductory computer programming course is to develop students' skills, so that they can solve problems through programming. Some studies showed that the students mastered the debugging skills developed better code writing skills [12, 15, 16]. The character metaphor as a variable

state helps to improve together the debugging and writing skills.

3) What features are missing to better support introductory programming learning?

We will list some ideas based on what we found and on common goals of introductory computer programming courses:

- We found only two games allowing more than one programming language. We suggest making the compiler and execution motor extensible and adaptable to increase the applicability of games in more contexts. This also increases the useful life of the game, as it may follow the trends in programming languages;
- We found no games that adapt their interaction according to students' evolution. For example, some games adopt the simplicity of block manipulation (visual languages), very useful in the beginning of learning, but insufficient at later stages. In other games the player has to write code in a programming language, demanding some level of domain of abstraction, notional machine and the language syntax and semantic, which can be too much for students in initial learning stages [2, 9, 11]. We suggest starting the game using block manipulation and evolving to code the programs as the student progresses into the game;
- In the same way, we did not find games that support learning during the gameplay experience. They only give feedback after the mission ends. If the game provides support while the learner is engaged in the task, it will help the student reflect about his/her errors in the context they happen [28].

Half of the games found in our research were described in the literature, but no version is available to use. The consequence is the constant development of new games with almost the same features. These works are often result of academic works, sometimes incomplete or with some gaps and bugs. This makes authors not wanting to make them available. Thus, we suggest open sourcing the tools into some online version control repository and promoting its reutilization, speeding up research on digital games applied to computer programming learning.

ACKNOWLEDGMENT

AV acknowledges the doctoral scholarship supported by CNPq/CAPES – Programa Ciência sem Fronteiras – CsF (6392-13-0) and authorized retirement by UDESC (688/13).

REFERENCES

[1] Barnes, T., Powell, E., Chaffin, A., Godwin, A. and Richter, H. "Game2Learn: Building CS1 Learning Games for Retention". June 2007. Proceedings of the 12th SIGCSE Conference on Innovation and Technology in Computer Science Education. Dundee, Scotland: p. 121–125.

[2] Gomes, A. and Mendes, A.J.N. "Learning to program-difficulties and solutions". September 2007. Proceedings of the International Conference on Engineering Education. Coimbra, Portugal.

[3] Aldrich, C. 2009. Learning Online with Games, Simulations, and Virtual Worlds: Strategies for Online Instruction. San Francisco: John Wiley & Sons.

[4] Bayliss, J.D. and Strout, S. "Games as a "flavor" of CS1". March 2006. Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education. Houston, Texas: p. 500–504.

[5] Leutenegger, S. and Edgington, J. "A games first approach to teaching introductory programming". March 2007. Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education. Covington, Kentucky: p. 115–118.

[6] Muratet, M., Torguet, P., Viallet, F. and Jessel, J. "Experimental feedback on Prog & Play, a serious game for programming practice". May 2010. Proceedings of the 31st Annual Conference of the European Association for Computer Graphics. Norrköping, Sweden: p. 1–8.

[7] Lee, M.J. and Ko, A.J. "Personifying Programming Tool Feedback Improves Novice Programmers' Learning". August 2011. Proceedings of the 7th International Workshop on Computing Education Research. Providence, Rhode Island: p. 109–116.

[8] Pea, R.D., Soloway, E. and Spohrer, J.C. 1987. "The buggy path to the development of programming expertise". Focus on Learning Problems in Mathematics. Vol. 9 (1), p. 5–30.

[9] Robins, A., Rountree, J. and Rountree, N. 2003. "Learning and teaching programming: A review and discussion". Computer Science Education. Vol. 13 (2), p. 137–172.

[10] Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M. and Paterson, J. "A Survey of Literature on the Teaching of Introductory Programming". June 2007. Proceedings of the 12th SIGCSE Conference on Innovation and Technology in Computer Science Education. Dundee, Scotland: p. 204–223.

[11] Boulay, B. du 1986. "Some difficulties of learning to program". Journal of Educational Computing Research. Vol. 2 (1), p. 57–73.

[12] Winslow, L.E. September 1996. "Programming pedagogy - a psychological overview". ACM SIGCSE Bulletin. Vol. 28 (3), p. 17–22.

[13] Mannila, L., Peltomäki, M. and Salakoski, T. September 2006. "What about a simple language? Analyzing the difficulties in learning to program". Computer Science Education. Vol. 16 (3), p. 211–227.

[14] Rosbach, A.H. and Bagge, A.H. "Classifying and Measuring Student Problems and Misconceptions". November 2013. Proceedings of the Norsk Informatikkonferanse. Stavanger, Norway.

[15] Lopez, M., Whalley, J., Robbins, P. and Lister, R. "Relationships between reading, tracing and writing skills in introductory programming". 2008. Proceedings of the 4th International Workshop on Computing Education Research. New York, New York, USA: p. 101–112.

[16] Lister, R., Fidge, C. and Teague, D. "Further evidence of a relationship between explaining, tracing and writing skills in introductory programming". August 2009. Proceedings of the 14th ACM SIGCSE Conference on Innovation and Technology in Computer Science Education. Paris-France: p. 161–165.

[17] Ahmadzadeh, M., Elliman, D. and Higgins, C. "An Analysis of Patterns of Debugging Among Novice". 2005. Proceedings of the 10th SIGCSE Conference on Innovation and Technology in Computer Science Education. : p. 84–88.

[18] Kramer, J. 2007. "Is abstraction the key to computing?". Communications of the ACM. Vol. 50 (4), p. 37–42.

[19] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.-D., Laxer, C., Thomas, L., Utting, I. and Wilusz, T. "A multi-national, multi-institutional study of assessment of programming skills of first-year CS students". June 2001. Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education. Canterbury, UK: p. 125–140.

[20] Lahtinen, E., Ala-Mutka, K. and Järvinen, H.-M. "A study of the difficulties of novice programmers". June 2005. Proceedings of the 10th SIGCSE Conference on Innovation and Technology in Computer Science Education. Monte de Caparica, Portugal: p. 14–18.

[21] ACM and IEEE 2013. Computer Science Curricula 2013.

[22] Hertz, M. "What do CS1 and CS2 mean?: investigating differences in the early courses". March 2010. Proceedings of the 41st ACM Technical Symposium on Computer Science Education. Milwaukee, USA: p. 199–203.

[23] Paliokas, I., Arapidis, C. and Mpimpitsos, M. "PlayLOGO 3D: A 3D interactive video game for early programming education: Let LOGO be a game". May 2011. Proceedings of the 3rd International Conference on Games and Virtual Worlds for Serious Applications. Athens: p. 24–31.

[24] Wassila, D. and Tahar, B. "Using serious game to simplify algorithm learning". July 2012. Proceedings of the International Conference on Education and e-Learning Innovations. Sousse: p. 1–5.

[25] Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M. and Khalil, M. April 2007. "Lessons from applying the systematic literature review process within the software engineering domain". The Journal of Systems and Software. Vol. 80 (4), p. 571–583.

[26] Kazimoglu, C., Kiernan, M., Bacon, L. and Mackinnon, L. January 2012. "A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming". Procedia - Social and Behavioral Sciences. Vol. 47, p. 1991–1999.

[27] Whitton, N. 2010. Learning with Digital Games: A Practical Guide to Engaging Students in Higher Education. Taylor & Francis.

[28] Van Merrienboer, J.J.G. and Kirschner, P.A. 2013. Ten steps to complex learning: a systematic approach to four-component instructional design. Taylor & Francis.

[29] Dim, C.A. and Edson, F. "APIN: Uma Ferramenta Para Aprendizagem de Lógicas e Estímulo do Raciocínio e da Habilidade de Resolução de Problemas em um Contexto Computacional no Ensino Médio". July 2011. Proceedings of the XIX Workshop sobre Educação em Computação. XXXI Congresso da Sociedade Brasileira de Computação. Natal, Brazil.

[30] Kahwage, C., França, E.L. de, Nunes, R.C., Carvalho, R. and Souza, D.T. "Jogo Baralho das Variáveis". July 2013. Proceedings of the XXXIII Congresso da Sociedade Brasileira de Computação. Maceio, Brazil: p. 450–459.

[31] Malliarakis, C., Satratzemi, M. and Xinogalos, S. "Towards a new massive multiplayer online role playing game for introductory programming". September 2013. Proceedings of the 6th Balkan Conference in Informatics. Thessaloniki, Greece: p. 156–163.

[32] Piteira, M. and Haddad, S. "Innovate in your program computer class: an approach based on a serious game". July 2011. Proceedings of the Workshop on Open Source and Design of Communication. Lisbon, Portugal: p. 49–54.

[33] Chang, J.K.-W., Dang, L.H., Pavleas, J., McCarthy, J.F., Sung, K. and Bay, J. 2012. "Experience with Dream Coders: developing a 2D RPG for teaching introductory programming concepts". Journal of Computing Sciences in Colleges. Vol. 28 (1), p. 227–236.

[34] Chaffin, A., Doran, K., Hicks, D. and Barnes, T. "Experimental Evaluation of Teaching Recursion in a Video Game". August 2009. Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games. New Orleans, Lousiana: p. 79–86.

[35] Scaico, P., Marques, D.L., Melo, L.D.A., André, M., Neto, S.V.M., Oliveira, A., Júnior, J.A., Labanca, M. and Scaico, A. "Um jogo para o ensino de programação em Python baseado na taxonomia de Bloom". July 2012. Proceedings of the XX Workshop sobre Educação em Computação. XXXII Congresso da Sociedade Brasileira de Computação. Curitiba, Brazil.

[36] Adamo-Villani, N., Cooper, S. and Whittinghill, D. "Building a serious game for teaching secure coding in introductory programming courses". May 2012. Proceedings of the 33rd Annual Conference of the European Association for Computer Graphics. Cagliari, Italy.

[37] Dantas, V.F., Alencar, L.A. and Freitas, P.J.B. "ProGame□: Um jogo para apoiar o ensino-aprendizagem de programação". 2011. Proceedings of the XXII Simpósio Brasileiro de Informática na Educação. Aracaju, Brazil.

[38] Coelho, A., Kato, E., Xavier, J. and Gonçalves, R. "Serious game for introductory programming". September 2011. Proceedings of the 2nd International Conference on Serious Games Development and Applications. Lisbon, Portugal: p. 61–71.

[39] Dantas, V.F., Macedo, E.R. De, Andrade, J.R.B., Coutinho, D.R.A., Cavalcante, A.F., Vasconcelos, T.G. and Pereira, M.E.D.S. "Combinando desafios e aventura em um jogo para apoiar a aprendizagem de programação em vários níveis cognitivos". 2013. Proceedings of the II Congresso Brasileiro de Informática na Educação. Campinas, Brazil.

[40] Wahner, T., Kartheuser, M., Sigl, S., Nolte, J. and Hoppe, A. "Logical Thinking by Play Using the Example of the Game “Space Goats”". September 2012. Proceedings of the 3rd International Conference on Serious Games Development and Applications. Bremen, Germany: p. 174–182.

[41] Barbosa, L.S., Fernandes, T.C.B. and Campos, A.M.C. "Takkou□: Uma Ferramenta Proposta ao Ensino de Algoritmos". July 2011. Proceedings of the XIX Workshop sobre Educação em Computação. XXXI Congresso da Sociedade Brasileira de Computação. Natal, Brazil.

[42] Maragos, K. and Grigoriadou, M. 2011. "Exploiting TALENT as a Tool for Teaching and Learning". The International Journal of Learning. Vol. 18 (1), p. 431–440.

[43] Liu, C.-C., Cheng, Y.-B. and Huang, C.-W. November 2011. "The effect of simulation games on the learning of computational problem solving". Computers & Education. Vol. 57 (3), p. 1907–1918.

[44] Zapušek, M. and Rugelj, J. March 2013. "Learning programming with serious games". EAI Endorsed Transactions on Game-Based Learning. Vol. 13 (03), p. 1–8.

[45] Eagle, M. and Barnes, T. "Experimental Evaluation of an Educational Game for Improved Learning in Introductory Computing". March 2009. Proceedings of the 40th ACM Technical Symposium on Computer Science Education. Chattanooga, USA: p. 321–325.