A Review of Serious Games for Programming

Michael A. Miljanovic and Jeremy S. Bradbury

University of Ontario Institute of Technology
2000 Simcoe St N, Oshawa, ON L1H 7K4, Canada
{michael.miljanovic,jeremy.bradbury}@uoit.ca

Abstract. A large number of games are available to students and instructors that aid in developing a basic understanding of how to read and write programs. In this paper we review the existing serious programming game literature and examine the educational content and game evaluations of 49 games. First, we assess all games with respect to the programming fundamentals specified in the ACM 2013 Computer Science Curricula guidelines. Next, we review how each game is evaluated with respect to likability, accessibility, learning effect and engagement. In addition to the evaluated research questions, we also review the research methods used in the evaluations. Based on the results of our survey we conclude by identifying a number of open problems in the serious programming games literature.

Keywords: Computer Science · Education · Game Evaluation · Programming · Serious Games · Survey · Systematic Review.

1 Introduction

Within the field of game-based Computer Science learning, a large number of games have been developed that focus on computer programming [55]. Unfortunately, serious programming games are often developed independently; existing work does not focus on methods that improve gameplay, and there is a need to analyze the use of games to support introductory programming [27].This means games may be created without learning from existing games, especially if games are not available via open-source licensing or other methods. Additionally, many serious games that claim to have positive outcomes for players lack any scientific validation [44]. Without any supporting evidence, it is difficult to compare serious games and judge which are the most effective learning tools.

In this paper we survey 49 serious programming games with respect to both game content and evaluation. Specifically, we survey these games to answer the following research questions:

- *What programming knowledge is covered by existing serious games?*
- *How are serious programming games evaluated?*

The games included in our survey are exclusively games that involve reading and/or writing programs in order to help players develop computer programming skills. The goal of our review is to provide a comprehensive overview of the

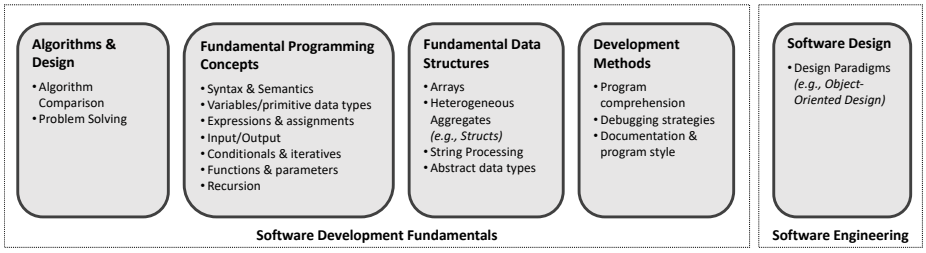


Fig. 1. ACM Computer Science Curricula 2013 knowledge areas [7]

state-of-the-art research in these serious programming games while also identifying open research problems. The open problems we identified fall into two categories: new opportunities for serious games development and new opportunities for enhancing evaluation best practices. Our review methodology is described in Section 2 followed by our review results in Section 3. Related work is discussed in Section 4. Finally, we discuss our results and present open problems in Section 5.

2 Methodology

2.1 Identification and Selection Criteria

We used different selection criteria for our categories of research and commercial games. The search term we used for both was "introductory programming games". Research games were first gathered using the top 200 results from a Google Scholar web search in English, then pruned based on our exclusion criteria. We then selected games that were discussed in the related works sections of those papers that fit our criteria. For inclusion in the study, research games must have an associated peer-reviewed paper or be the subject of a thesis project. Games developed by researchers with no oversight were excluded. Unfortunately, not all research games were available to be played and we therefore had to evaluate some based only on their descriptions and not a first-hand evaluation. Commercial games were collected using a Google web search and through online game stores[1]. This includes games like Code Hunt [53], which is a game developed by Microsoft Research, but has been discussed in several peer-reviewed papers.

2.2 Classification Criteria

Audience. The targeted audiences for serious programming games is broken down into four separate categories:

[1] Commercial sources such as the iOS game store, Google Play Store, HourOfCode.com, and Tynker.com offer over 100 different programming games, typically aimed at children. However, many of these games follow a similar approach or share visual programming environments. We chose to only include a small sample from these websites due to the high degree of game overlap and similarity.

- Children (age 5-13)
- High school students (age 14-17)
- Undergraduate novices (age 18+, no programming experience)
- Undergraduate adepts (age 18+, programming experience)

Educational Content. The 2013 ACM Curriculum Guidelines for Undergraduate Degree Programs in Computer Science [7] include areas of knowledge for students learning programming (see Figure 1). The concepts we selected were from the Tier 1 list of knowledge areas for computer science, meaning that the topics are intended to be introduced to students in their first or second years of study at the undergraduate level.

We focused on the Software Development Fundamentals (SDFs) and Software Engineering (SE) knowledge areas. SDFs are critical for students to become both competent at programming and knowledgeable about designing and analyzing algorithms. The other software-oriented knowledge areas discussed in the ACM Curriculum require students to have strong foundations in SDFs. In addition, a number of games have been developed to help students understand the object-oriented paradigm; we chose to include this specific section of Software Engineering to acknowledge the multitude of games that included or focused on object-oriented development. Due to space limitations, we chose to exclude knowledge areas and concepts that were not included in more than two games.

We differentiate between the inclusion of educational content with a primary focus versus a secondary focus. Educational content that is classified as being a primary focus of a given serious game indicates that the game designers emphasize that their game is designed to teach that content. In cases where the primary focus is not explicitly stated, we make a determination based on play testing or on a description of the game play. Alternatively, if educational content is present in a serious game but not emphasized, we classify this as having a secondary focus. Content with a secondary focus may be introduced in the game or may need to be learned prior to playing.

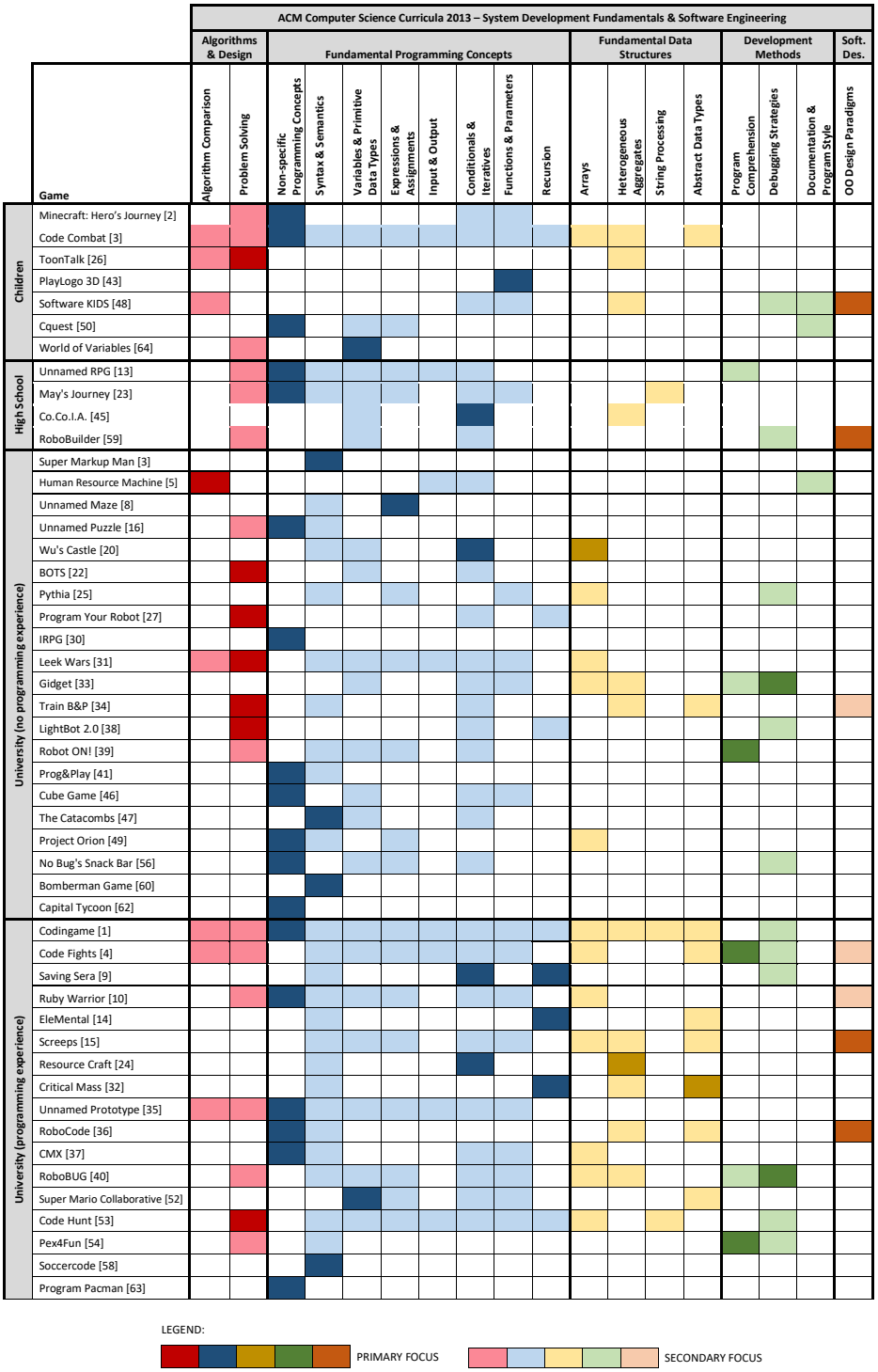
Evaluation. When available, we also review and classify the evaluation of a serious game[2]. First, we identify which games evaluate learning outcomes, player engagement, positive feedback, and accessibility. Second, we classify the evaluation methods used, ranging from informal player feedback and game results to full empirical studies about learning outcomes.

3 Results

We identified 49 serious programming games that met our selection criteria – 36 research games and 13 commercial games. Approximately half (23) of these games can be downloaded or played online.

[2] Serious games with evaluations are primarily a subset of those that have accompanying research papers, technical reports or theses.

Table 1. Classification of serious programming games based on educational content



3.1 Audience

The largest audience of the surveyed games was undergraduate novices with no programming experience (21 games) followed by undergraduates with some programming experience (17 games), children (seven games) and high school students (four games).

3.2 Educational Content

The educational content in each game was assessed based on the knowledge areas identified in the ACM Computer Science Curricula 2013 (see Table 1):

- **Algorithms and Design:** this SDF unit covers the importance of algorithms in problem-solving, including mathematical functions and divide-and-conquer strategies. The comparison of algorithms in the surveyed serious games was not widely covered. An exception was the Human Resource Machine game in which players are incentivized to minimize the number of instructions and steps taken to complete tasks. When algorithm comparison was included, it was often done informally and without any mention of algorithmic complexity. Interestingly, only 21 of the 49 games had an emphasis on problem solving. For example, Robocode [36] is not problem solving-based but is instead competition-based with players completing programming challenges against opponents.
- **Fundamental Programming Concepts:** these are the most commonly targeted topics for serious programming games, which is consistent with the goal of introducing students to programming and helping them learn how to read and write code. 'Syntax and Semantics' was the most commonly covered concept with 30 games using some sort of written programming language; the remaining games used a drag-and-drop block interface for creating programs, or use a high-level language with little room for error (e.g. Gidget [33]). The next most widely covered concept was 'conditionals and iteratives' with 28 games, followed by 'variables and primitive data types' with 23 games. Not all games required fundamental concepts like variables. For example, PlayLogo 3D [43] does not include variables as players need only submit individual commands with functions to play the game. While 'Recursion' was the primary focus of several games [9, 14, 32], it was one of the least covered concepts along with 'Input and Output.' This result was surprising given that the target audience for many of the games was university students with some programming experience.
- **Fundamental Data Structures:** the most common data structure concepts were 'arrays and lists' (13 games) and 'heterogeneous structures' (12 games). Few research papers explicitly stated a focus on data structures, and our identification was primarily the result of game testing and reading game play descriptions. 'Abstract data types' were not included in most games. Exceptions include Critical Mass [32], which required players to navigate a tree structure. Finally, 'string processing' was only a secondary focus of

three games, and other ACM concepts including 'linked lists' and 'referencing' were not targeted by any of the games.
- **Development Methods:** 'debugging' was the most commonly targeted development concept, with 12 games featuring some focus on debugging code. However, this does not include all of the ACM's program correctness topics (e.g. test-case generation, unit testing). Even the games that choose to focus on debugging [40, 33] are not comprehensive with respect to debugging topics. 'Program comprehension' was the focus of a few games, but the vast majority of games required players to write their own code. CodeFights is an example of a game where players must interpret code written by someone else and develop program comprehension skills through trying to understand foreign code. Although games with real programming languages allow for commenting, very few games focused on documentation and program style, and only did so as a secondary focus. Other development methods, including refactoring and the use of software libraries, were not covered by the games.
- **Software Design:** the majority of software design areas presented by the ACM are intended for learners above the beginner level. However, the curriculum indicates that software design should be covered at an early stage. IBM's Robocode [12, 21, 36, 42] has a strong focus on software design – specifically object-oriented (OO) design, as players learn about abstraction through the use and modification of the game's robot objects.

3.3 Learning Focus

Identifying the primary focus of serious programming games was especially difficult when not explicitly stated by the game designers. When not stated, we based our identification of a primary focus from playing the available games and inferring based on the content of the research papers. In the end we found that 18 games focused primarily on general introductory programming, without a specific topic. Most of these 18 games included other fundamental programming concepts, but there were some research papers that introduced a game for learning introductory programming without detailing specific content. Problem solving was the second most common focus, with 7 different games. One example of this is Lightbot 2.0 [38], where players do not learn a programming language but do develop an understanding of sequencing and implementation of algorithms. The general trend of programming games that focus on problem solving is that they target simple problems and program-based solutions, with limited or no emphasis on formality.

3.4 Evaluation

A variety of evaluation methods were used in the surveyed games (see Table 2) – 23 surveys, 11 sets of game play statistics, 10 skill tests, four sets of interviews and one evaluation using expert feedback. 21 games used only one evaluation method while 14 used multiple methods. The most common evaluation subject matter was positive feedback. There were 21 cases of participants reporting that

Table 2. Classification of serious programming games based on evaluations methods

		Research Questions				Method of Evaluation					
	Game	Did the users have positive feelings about the game?	Was the game accessible?	Were users engaged while playing the game?	Was there a learning effect from playing the game?	Informal Feedback	Survey/Questionnaire	Formal Interview	Skill Tests	Game Play Statistics	Expert Feedback
Children	ToonTalk [26]		✓							●	
	PlayLogo 3D [43]		✓								●
	Software KIDS [48]	✓					●				
	Cquest [50]	✓				●					
High School	Unnamed RPG[13]				✓		●		●		
	May's Journey [23]	✓					●			●	
	Co.Co.I.A. [45]			✓		●					
	RoboBuilder [59]			✓		●					
University (no programming experience)	Unnamed Maze [8]			✓			●			●	
	Unnamed Puzzle [16]			✓			●				
	Wu's Castle [20]				✓		●		●		
	BOTS [22]				✓				●		
	Pythia [25]		✓				●				
	Program Your Robot [27]	✓					●				
	IRPG [30]	✓				●					
	Gidget [33]			✓	✓				●	●	
	Train B&P [34]			✓	◆		●			●	
	LightBot 2.0 [38]	✓			✓		●				
	Robot ON! [39]	✓			✓		●	●	●		
	Prog&Play [41]	✓	✓		◆		●		●	●	
	The Catacombs [47]	✓	✓	✓	◆		●	●	●	●	
	Project Orion [49]	✓	✓		◆		●				
	No Bug's Snack Bar [56]	✓				●					
	Capital Tycoon [62]			✓	◆		●				
University (programming experience)	Saving Sera [9]	✓	✓	✓	◆		●	●	●	●	
	EleMental [14]	✓			✓		●		●	●	
	Resource Craft [24]				✓		●			●	
	Critical Mass [32]	✓					●			●	
	Unnamed Prototype [35]			✓				●			
	RoboCode [36]	✓		✓	✓		●				
	CMX [37]	✓					●				
	RoboBUG [40]	✓	✓		◆		●		●		
	Code Hunt [53]	✓				●					
	Pex4Fun [54]	✓				●					
	Soccercode [58]	✓				●					
	Program Pacman [63]	✓					●				

EMPIRICAL EVIDENCE FOR A GIVEN RESEARCH QUESTION WAS POSITIVE (✓) or INCONCLUSIVE (◆)
DATA WAS COLLECTED & ANALYZED USING A GIVEN RESEARCH METHOD (●)

they liked a game, often through a survey. 16 games were evaluated for learning effects on the players, but unfortunately seven of these did not have a statistically significant learning effect. Although many of the papers cited engagement as a motivation for using serious games, only 11 were actually evaluated for player engagement. Finally, only eight of the games were tested for accessibility.

4 Related Work

Although there are reviews that investigate the impact of serious games [17, 19], there is very little research focusing on serious programming games. One exception is a review by Vahldick et al. [55], that focuses specifically on games for improving introductory programming skills. The review categorizes 40 games by type (Logo-based, adventure, general), platform (Windows, iOS, Java, Web, Android, Linux), competency (writing, reading, debugging), topic (including some of the ACM 2013 CS curricula topics from SDFs), and language (Textual/visual block graphics, Java/Javascript, C/C++/C#, and others).

Our work has two similarities with the Vahldick et al. review – first, 15 games are included in both studies and second, both studies survey the learning topics or content of the games. With regard to this overlap, we have included 34 games in our study that were not included by Vahldick et al. Furthermore, 25 games included in their study were not included in ours. Reasons for exclusion include: 13 games were outside of our selection criteria (e.g., non-english, not focused on learning programming), four games were extremely similar to other games in the survey, and eight games were no longer available online and did not have published papers. Our initial intention was to include as many of the previously studied games as possible in order to reproduce and validate the learning portion of the Vahldick et al. results. However, this was not possible as many of the overlapping games have been updated in the three years since their study and we no longer have access to the versions of games surveyed. The main difference between our work and the Vahldick et al. review is that we have surveyed a wider selection of games with the intention of assessing the learning aspects of the games (learning content and learning evaluation) as opposed to the game characteristics (e.g., platform, language, genre).

5 Discussion & Conclusion

Our results show that the 49 serious programming games surveyed focus primarily on a subset of the ACM computing knowledge areas. Unfortunately, many of the games are not released publicly and we were unable to independently verify the learning content of these games through play testing. The lack of access is problematic for both researchers developing computer science educational games and instructors seeking to find effective learning tools. The surveyed serious games focus largely on the problem solving and fundamental programming concepts knowledge areas. There are a lack of games that focus on data structures, development methods and software design. Furthermore, while the

primary learning focus of many games was introductory programming, few of the games appear to cover all of the ACM's SDFs. This indicates **a need to determine if new serious programming games can bridge the curricula gaps**. It is possible that some SDFs are not well suited for game-based learning.

With respect to game design, we observed that the majority of the games were not multiplayer. There is **a need for further research on the learning benefits of competitive and collaborative serious games for programming**. We also observed that while a number of games were designed with accessibility and inclusivity in mind (e.g. Saving Sera [9], May's Journey [23]), many did not include any detail on these important aspects of design. This maybe an indicator that **best practices for accessible and inclusive design of serious programming games need to be adopted.**

With respect to the evaluation of serious program games, we were unable to observe common methodological practices other than a tendency to assess if players liked a game. This indicates a **need for the establishment of best practices in evaluating serious programming games**. We believe that the use of alternative evaluation methods in addition to in-class studies would be beneficial. In particular, it may be helpful to consider controlled experiments and expert feedback (e.g. used only in PlayLOGO 3D [43]) in combination with playability heuristics [18]. Finally, in addition to establishing best practices for evaluation, there is a **need for third-party evaluations**. Third-party evaluations do not suffer from self-confirmatory bias, provide valuable data that can independently validate a serious game's learning effects, and can lead to wider adoption of serious games in Computer Science education.

Acknowledgment

This research was partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Codingame: Coding games and programming challenges to code better. https://www.codingame.com/ (2012), accessed: 2017-04-12
2. CodeCombat: Learn how to code by playing a game. https://codecombat.com/ (2013), accessed: 2017-04-12
3. Super Markup Man. http://store.steampowered.com/app/502210/ (2013), accessed: 2017-04-12
4. Codefights: Test your code. https://codefights.com/ (2015), accessed: 2017-04-12
5. Human Resource Machine. https://tomorrowcorporation.com/humanresourcemachine (2015), accessed: 2017-04-12
6. Minecraft: Hour of code tutorials. https://code.org/minecraft (2017), accessed: 2017-04-12
7. ACM/IEEE-CS Joint Task Force on Computing Curricula: Computer science curricula 2013. Tech. rep., ACM Press and IEEE Computer Society Press (Dec 2013)

8. Adamo-Villani, N., Haley-Hermiz, T., Cutler, R.: Using a serious game approach to teach 'operator precedence' to introductory programming students. In: Proc. of the Int. Conf. on Information Visualisation (IV 2013). pp. 523–526 (2013)
9. Barnes, T., Powell, E., Chaffin, A., Lipford, H.: Game2Learn : Improving the motivation of CS1 students. In: Proc. of the 3rd Int. Conf. on Game Development in Computer Science Education (GDCSE '08). pp. 1–5 (2008)
10. Bates, Ryan: Ruby Warrior: Popular free Ruby programming tutorial game. https://www.bloc.io/ruby-warrior/ (2010), accessed: 2017-04-13
11. Begel, A.: LogoBlocks: A graphical programming language for interacting with the world. Master's thesis, Electrical Engineering and Computer Science, MIT (1996)
12. Bonakdarian, E., White, L.: Robocode throughout the curriculum. Journal of Computing Sciences in Colleges **19**(3), 311–313 (2004)
13. Buckley, C.: Design and Implementation of a Genre Hybrid Video Game that Integrates the Curriculum of an Introductory Programming Course. Master's thesis, Clemson University (2012)
14. Chaffin, A., Doran, K., Hicks, D., Barnes, T.: Experimental evaluation of teaching recursion in a video game. In: Proc. of the 2009 ACM SIGGRAPH Symp. on Video Games. vol. 1, pp. 79–86 (2009)
15. Chivchalov, A., Chivchalov, A., Gunyakov, S.: Screeps. https://screeps.com/ (2016), accessed: 2017-04-12
16. Coelho, A., Kato, E., Xavier, J., Goncalves, R.: Serious game for introductory programming. Lecture Notes in Computer Science **6944 LNCS**, 61–71 (2011)
17. Connolly, T.M., Boyle, E.A., MacArthur, E., Hainey, T., Boyle, J.M.: A systematic literature review of empirical evidence on computer games and serious games. Computers & Education **59**(2), 661–686 (2012)
18. Desurvire, H., Wiberg, C.: Game usability heuristics (play) for evaluating and designing better games: The next iteration. In: Int. Conf. on Online Communities and Social Computing. pp. 557–566. Springer (2009)
19. Dondlinger, M.J.: Educational video game design: A review of the literature. Journal of Applied Educational Technology **4**(1), 21–31 (2007)
20. Eagle, M., Barnes, T.: Experimental evaluation of an educational game for improved learning in introductory computing. In: Proc. of the 40th ACM Technical Symp. on Computer science education (SIGCSE'09). pp. 321–325 (2009)
21. Hartness, K.: Robocode: using games to teach artificial intelligence. Journal of Computing Sciences in Colleges **19**(4), 287–291 (2004)
22. Hicks, A.: Towards social gaming methods for improving game-based computer science education. In: Proc. of the Fifth Int. Conf. on the Foundations of Digital Games - FDG '10. pp. 259–261 (2010)
23. Jemmali, C., Zijian, Y.: May's Journey: A serious game to teach middle and high school girls programming. Master's thesis, Worcester Polytechnic Institute (2016)
24. Jiau, H.C., Chen, J.C., Ssu, K.F.: Enhancing self-motivation in learning programming using game-based simulation and metrics. IEEE Transactions on Education **52**(4), 555–562 (2009)
25. Johnsen, A.L., Ushakov, G.: Python programming game. Master's thesis, Norwegian University of Science and Technology (2011)
26. Kahn, K.: A computer game to teach programming introduction to ToonTalk. In: Proc. of the National Educational Computing Conf. pp. 127–135 (1999)
27. Kazimoglu, C., Kiernan, M., Bacon, L., Mackinnon, L.: A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. vol. 47, pp. 1991–1999. Elsevier (2012)

28. Kelleher, C., Cosgrove, D., Culyba, D., Forlines, C., Pratt, J., Pausch, R.: Alice2: programming without syntax errors. In: Proc. of 15th Int. Symp. on User Interface Software and Technology (UIST'02) - Demonstrations. pp. 35–36 (2002)
29. Kelly, J.O., Maynooth, N.U.I., Gibson, J.P.: RoboCode & Problem-Based Learning: A non-prescriptive approach to teaching programming. ACM SIGCSE Bulletin **38**(3), 217–221 (2006)
30. Khenissi, M., Essalmi, F., Jemni, M.: Presentation of a learning game for programming languages education. In: Proc. of IEEE 13th Int. Conf. on Advanced Learning Technologies (ICALT 2013). pp. 324–326 (2013)
31. Laupretre, Pierre: Leek Wars. https://leekwars.com/ (2016), accessed: 2017-04-12
32. Lawrence, R.: Teaching data structures using competitive games. IEEE Transactions on Education **47**(4), 459–466 (2004)
33. Lee, M.J., Ko, A.J.: Investigating the role of purposeful goals on novices' engagement in a programming game. In: Proc. of IEEE Symp. on Visual Languages and Human-Centric Computing, VL/HCC. pp. 163–166 (2012)
34. Liu, C.C., Cheng, Y.B., Huang, C.W.: The effect of simulation games on the learning of computational problem solving. Comp. and Edu. **57**(3), 1907–1918 (2011)
35. Ljungkvist, P., Mozelius, P.: Educational games for self learning in introductory programming courses - a straightforward design approach with progression mechanisms. In: Proc. of the 6th European Conf. on Games Based Learning. pp. 285–293 (2012)
36. Long, J.: Just for fun: Using programming games in software programming training and education – A field study of IBM Robocode community. Journal of Information Technology Education **6**, 279–290 (2007)
37. Malliarakis, C., Satratzemi, M., Xinogalos, S.: CMX: Implementing an MMORPG for learning programming. In: Proc. of the 8th European Conf. on Games Based Learning (ECGBL2014). pp. 346–355 (2014)
38. Mathrani, A., Christian, S., Ponder-Sutton, A.: PlayIT: Game based learning approach for teaching programming concepts. Educational Technology & Society **19**(5), 5–17 (2016)
39. Miljanovic, M.A., Bradbury, J.S.: Robot ON!: a serious game for improving programming comprehension. In: Proc. of the 5th Int. Workshop on Games and Software Engineering. pp. 33–36 (2016)
40. Miljanovic, M.A., Bradbury, J.S.: RoboBUG: A serious game for learning debugging techniques. In: Proc. of the 2017 ACM Conference on International Computing Education Research (ICER '17). pp. 93–100 (2017)
41. Muratet, M., Delozanne, E., Torguet, P., Viallet, F.: Serious game and students' learning motivation: effect of context using Prog&Play. In: Proc. of the 11th Int. Conf. Intelligent Tutoring Systems (ITS 2012). pp. 123–128 (Jun 2012)
42. O'Kelly, J., Gibson, J.P.: RoboCode & problem-based learning : A non-prescriptive approach to teaching programming. In: Proc. of Annual Conf. on Innovation and Technology in Comp. Sci. Education (ITiCSE '06). pp. 26–28 (2006)
43. Paliokas, I., Arapidis, C., Mpimpitsos, M.: Game based early programming education: The more you play, the more you learn. Lecture Notes in Computer Science **7544**, 115–131 (2013)
44. Pandeliev, V.T., Baecker, R.M.: A framework for the online evaluation of serious games. In: Proc. of the Int. Academic Conf. on the Future of Game Design and Technology. pp. 239–242. ACM (2010)
45. Pellas, N.: Exploring interrelationships among high school students engagement factors in introductory programming courses via a 3D multi-user serious game created in Open Sim. J. of Universal Computer Science **20**(12), 1608–1628 (2014)

46. Piteira, M., Haddad, S.R.: Innovate in your program computer class: an approach based on a serious game. In: Proc. of Workshop on Open Source and Design of Communication. pp. 49–54 (2011)
47. Ralph, T., Barnes, T.: The Catacombs : A study on the usability of games to teach introductory programming (2006)
48. Ramirez-Rosales, S., Vazquez-Reyes, S., Villa-Cisneros, J.L., De Leon-Sigg, M.: A serious game to promote object oriented programming and software engineering basic concepts learning. In: Proc. of 4th Int. Conf. in Software Engineering Research and Innovation (CONISOFT 2016). pp. 97–103 (2016)
49. Romo, E.K.: Game design for a serious game to help learn programming. Master's thesis, Faculdade de Engenharia da Universidade do Porto (2011)
50. Singh, J., Wei, L.L., Shanmugam, M., Gunasekaran, S.S., Dorairaj, S.K., Tenaga, U., Uniten, N.: Designing computer games to introduce programming to children. In: Proc. of 4th Int. Conf. on Info. Tech. and Multimedia. pp. 643–647 (2008)
51. Tessler, J., Beth, B., Lin, C.: Using cargo-bot to provide contextualized learning of recursion. In: Proc. of the 9th Annual Int. ACM Conf. on Internation Computing Education Research (ICER'13). pp. 161–168 (2013)
52. Theodorou, C., Kordaki, M.: Super Mario: a collaborative game for the learning of variables in programming. Int. Journal of Academic Research **2**(4), 111–118 (2010)
53. Tillmann, N., Bishop, J.: Code Hunt: searching for secret code for fun. In: Proc. of the 7th Int. Work. on Search-Based Software Testing (SBST'14). pp. 23–26 (2014)
54. Tillmann, N., Halleux, J.D., Bishop, J.: Teaching and learning programming and software engineering via interactive gaming. In: Proc. of the Int. Conf. on Software Engineering (ICSE 2013). pp. 1117–1126 (2013)
55. Vahldick, A., Mendes, A.J., Marcelino, M.J.: A review of games designed to improve introductory computer programming competencies. In: Proc. of the Frontiers in Education Conf. (FIE 2014). pp. 1–7. IEEE (2014)
56. Vahldick, A., Mendes, A.J., Marcelino, M.J.: Analysing the enjoyment of a serious game for programming learning with two unrelated higher education audiences. In: Proc. of the European Conf. on Games-based Learning. pp. 523–531 (2015)
57. Vassilev, T.I., Mutev, B.I.: An approach to teaching introductory programming using games. In: Proc. of the Int. Conf. on e-Learning. pp. 246–253 (2014)
58. Wang, M., Hu, X.: SoccerCode: A game system for introductory programming courses in computer science. In: Proc. of the World Congress on Engineering and Computer Science (WCECS 2011). pp. 282–287 (Oct 2011)
59. Weintrop, D., Wilensky, U.: RoboBuilder: Video game program-to-play constructionist. In: Proc. of Constructionism 2012. pp. 1–5 (2012)
60. Wong, W., Chou, Y.: An interactive Bomberman game-based teaching/learning tool for introductory C programming. In: Proc. of the Int. Conf. on Technologies for E-Learning and Digital Entertainment. pp. 433–444 (2007)
61. Yan, L.: Teaching object-oriented programming with games. In: Proc. of the 6th Int. Conf. on Information Technology: New Generations. pp. 969–974. IEEE (2009)
62. Yeh, K.c.M., Chen, W.f.: WIP: Using a computer gaming strategy to facilitate undergraduates' learning in a computer programming course : An experimental study. In: Proc. of the 41st ASEE/IEEE Frontiers in Edu. Conf. pp. 11–12 (2011)
63. Yue, W.S., Wan, W.L.: The effectiveness of digital game for introductory programming concepts. In: Proc. of the 10th Int. Conf. for Internet Technology and Secured Transactions (ICITST 2015). pp. 421–425 (2015)
64. Zapušek, M., Rugelj, J.: Learning programming with serious games. EAI Endorsed Transactions on Game-Based Learning **13**(1), 1–8 (2013)