Revista Brasileira de Informática na Educação – RBIE
Brazilian Journal of Computers in Education
(ISSN online: 2317-6121; print: 1414-5685)
http://br-ie.org/pub/index.php/rbie

Submission: 31/May/2021; 1st round notif.: 19/Jul/2021; New version: 22/Sep/2021; 2nd round notif.: 28/Oct/2021;
Camera ready: 10/Nov/2021; Edition review: 23/Nov/2021; Available online: 01/Dec/2021; Published: 01/Dec/2021;

Analyzing novices' fun and programming behaviors while playing a serious blocks-based game

Adilson Vahldick
UDESC
ORCID: 0000-0002-0442-3735
adilson.vahldick@udesc.br

Maria José Marcelino
Universidade de Coimbra
ORCID: 0000-0002-1989-5559
zemar@dei.uc.pt

António José Mendes
Universidade de Coimbra
ORCID: 0000-0001-6659-660X
toze@dei.uc.pt

Abstract
Blocks-based environments have been used to promote programming learning mostly in elementary and middle schools. In many countries, isolated initiatives have been launched to promote programming learning among children, but until now there is no evidence of widespread use of this type of environment in Brazil and Portugal. Consequently, it is common that many students reach higher education with little or no programming knowledge and skills. NoBug's SnackBar is a game designed to help promote programming learning. This study examined students' behavior and attitudes when playing the game on their initiative. It used a sample of 33 undergraduate students enrolled in an introductory programming course. The variables studied were students' performance and engagement, satisfaction, and problem-solving strategies. The main findings were (1) better performing students had a high level of perceived learning, (2) all the students had similar perceptions about their fun while playing, (3) the leader board was the most used game element not directly related to learning and (4) the top-ranked students access previous solutions to help them solve a new mission, while the others often use a trial-and-error approach.
Keywords: *computer programming learning; blocks-based approach; serious games.*

Cite as: Vahldick, A., Marcelino, M. J., & Mendes, A. J. (2021). Analyzing novices' fun and programming behaviors while playing a serious blocks-based game. Revista Brasileira de Informática na Educação, 29, 1337-1355. DOI: 10.5753/rbie.2021.2069

1 Introduction

New generation students, used to games and other electronic media, are not motivated by exercises to calculate and print numbers on the console or in a window but are used to consume animations, graphics and sounds, and probably these are the types of media they would like to produce (Razak et al., 2019). The use of games has been integrated with the curriculum to simulate real-life activities (Johnson et al., 2015) and to provide meaningful learning opportunities in the hope of increasing students' interest in educational content (Weintrop & Wilensky, 2016b). With games students can learn in a personalized way, games can adjust themselves according to the player, and in a self-supervised way, players are aware of their mistakes when they fail in the tasks of the game, which in turn instructs the player on how to perform a certain action (Prensky, 2001). Serious Games are designed to have instruction as the primary goal (Arnab et al., 2012), allowing students to develop new skills, learn new knowledge, and strengthen existing competencies (Boller & Kapp, 2017).

A good way to promote programming learning is to have a disciplined and intensive practice (Robins, 2019). However, students often lack the motivation to engage in programming tasks. Practicing problem-solving in games can be more motivating than using traditional exercises, as it promotes confidence through experience in building sets of solution patterns that will be very useful when students encounter problem-solving situations using real programming languages (Shabalina et al., 2017).

Vahldick et al. (2014) studied about 40 games designed to support programming learning. They concluded that most of the analyzed games had no game elements to promote extrinsic motivation through fun. Also, most games didn't include typical game elements, such as points or bonuses. However, Koster (2014) points out that earning points makes players more committed to the game, and Prensky (2001) mentions that fun allows students to accomplish their tasks more easily, even if they require more effort. Based on these ideas, we developed a serious game called NoBug's Snack Bar (Vahldick et al., 2020). The game includes some common elements, such as point scoring, player's leader boards, and even the customization of player's avatars, intending to increase the feeling of belonging, satisfaction, and motivation to overcome the challenges of the game (Mazlan & Burd, 2011).

One of the problems in introductory programming learning is syntax errors (Bosse & Gerosa, 2017). Therefore, like many of the games analyzed in Vahldick et al. (2014), a Block-Based Programming (BBP) approach was adopted, because the goal is not learning an actual programming language, but rather the development of computational problem-solving skills and competencies. Environments that follow this approach present program execution as animations (Sorva et al., 2013) and use graphical notations to produce solutions (Ben-Ari, 2013). The actions, variable manipulation, and control structures are represented by colored blocks that fit together following the Lego metaphor (Weintrop & Wilensky, 2016a). In these environments, the student should use a notation focused on logic and solution building than to be concerned with the language grammar rules (Kelleher & Pausch, 2005). An ideal environment for introductory programming learning should provide a simple interface that supports viewing objects, has a block-based editor, reports simple error messages and instructs how to correct them, and can execute a program step-by-step (Xinogalos et al., 2017).

The development of NoBug's Snack Bar was carried out within four iteration cycles, involving novice students, where they could play anytime and anywhere. This paper reports the last cycle where students were organized into two groups according to their performance. The research aimed to identify students' behavior and opinions while solving programming problems using this serious game. This led to the definition of four research questions:

RQ1. What is the difference between the groups regarding the feelings about their learning?

RQ2. What is the difference between the groups regarding the feelings about fun?

RQ3. What is the most used game element not directly related to learning?

RQ4. What is the student's behavior when solving problems in the game?

2 Pedagogical background

Computers can enhance learning when students can see the concrete results of their efforts (Papert, 1980). In the constructionist approach of education, students coordinate their learning by constructing, manipulating, and testing concepts in a microworld (Laurillard et al., 2013). A microworld is a space with assumptions and constraints that provides a context for the learner to construct knowledge through experimentation (Papert, 1980). Students learn by exploration and construction in this world, where the effects of their actions are reflected in what is correct or incorrect in their beliefs. Learning results occur through active practice.

Constructionist games bring ideas like student-directed learning, meaningful personal constructions, emphasizing meaningful ideas into their design (Weintrop & Wilensky, 2014). In the last decade, programming environments and games have materialized the constructionist learning approach. To learn abstract programming concepts, students need to build them by hands-on experience. Constructionist environments allow the development of two essential skills for programming learning: procedural reasoning and debugging. Thinking procedurally involves breaking a problem down into smaller parts and recognizing patterns that can actually repeat themselves (Papert, 1980). Debugging involves systematically trying to adjust a piece of code to identify and correct errors to keep the system running properly (Holbert & Wilensky, 2011).

Constructionist games have two design principles: their tools and resources must be expressive, and the goals need to encourage exploration (Weintrop et al., 2012). The size of the building blocks should allow students to express ideas and strategies that are meaningful in their learning context: not so large that the game is too easy, and not so small to avoid boredom or very difficult tasks. Games can reward a variety of findings and are not limited to a single or a small set of winning strategies. Creation activities can take many forms, but the resulting artifacts must be identifiable and useful. Furthermore, the typical interaction and response cycle for programming learning (Kazimoglu et al., 2013) is suitable for any constructionist game: (1) students develop, execute, or debug the solution, (2) the game performs the actions based on the submitted solution, and (3) the game provides the results, answers, and support to the student. This iterative and interactive cycle provides powerful possibilities for students to try, correct and repeat their attempts and improve their abilities.

3 Related work

In order to find other experiences involving games that have used the BBP approach in higher education, a literature review was carried out over the last five years publications (2016-2021). We only considered papers that report experiences in teaching programming in higher education. Some papers describe new components such as automated testing or automated feedback, but with experiments limited to component evaluation, not classroom experiments. Many papers focus on elementary and middle school students, while the number of papers related to higher education is much more limited. After analyzing the articles, only a small number was found that can be compared with the present study in the experimentation of the BBP approach in higher education. It can be noted that none of them were experienced in undergraduate courses in computer science.

Parsons problems are code-completion problems, which require students to rearrange mixed up blocks of code to create the correct solution to a programming problem. Zhi et al. (2019) presented a study that evaluates the effectiveness of Parsons problems for block-based programming. To investigate this impact, they designed and integrated Parsons problems into Snap![1]. The participants were non-STEM major undergraduate students, with minimal prior programming experience, enrolled in a CS0 course at a U.S. research university over 6 semesters (Fall 2016 through Spring 2019). The study analyzed 6 assignments. In the last semester, students solved assignments with Parsons problems, rather than traditional problem solving by writing code. To identify problem-solving behaviors, the authors investigated three potential unproductive behaviors: searching for blocks to use, editing block inputs, and testing irrelevant blocks. They concluded that although Parsons problems prevented these unproductive behaviors during the lab, they would not disproportionately increase these unproductive times during the homework. They found Parsons problems saved students a significant amount of total problem-solving time, without reducing performance on subsequent problems.

BlockPy is a block-based editor for the Python programming language (Bart et al., 2020). It is a dual block/text editor that allows students to switch at any time between a block or a text representation of their code. BlockPy has an embedded data science context, so inputs are the data available in the Python library (weather, stocks, earthquakes, crimes and books) and the outputs of the programs are graphs. BlockPy has been used in a non-Computer Science majors introductory Computational Thinking course for four semesters. Most students had no prior programming experience and a limited understanding of the field. In the first 6 weeks of the course, students created their algorithms using natural language and flowcharts. After they had 3 classes over 2 weeks where they used BlockPy in blocks mode, addressing common topics like variables, conditionals and loops. On the last day, students were encouraged to use text mode in order to become familiar with writing code in text format. The evaluation pointed out that many students remained almost entirely in blocks mode, even during the final problems, while a small number of students used text mode almost exclusively, having even expressed that they found the text interface more understandable than the blocks interface. The authors also found that many students got confused in the transition from BlockPy text mode and Python.

Crescendo, is a self-paced programming practice environment that combines the block-based and visual, interactive programming of Snap!, with structured practices commonly found in Drill-and-Practice Environments (Wang et al., 2020). It organizes small programming tasks into challenges based on programming concepts such as loops and conditionals. Each concept may have multiple challenges and each challenge covers a single learning objective of a concept. In each challenge, students accomplish three programming tasks following the Use-Modify-Create (UMC) scaffolding (Lee et al., 2011). Crescendo was used to implement two mandatory and one optional challenge focused on loops. The study involved 50 students enrolled in an undergraduate CS0 course. Within each challenge, progression tasks were designed following the UMC framework that provided engaging and objective activities with a slowly increasing level of difficulty. The main conclusions of this work were that interactive programs can maintain engagement even when problems are small and objective. Also, the system immediate feedback allows students to progress independently.

[1] https://snap.berkeley.edu/

4 Materials and methods

4.1 Research methodology

The game research and development process followed the steps indicated in Figure 1. This model is an adaptation of Design-Based Research (DBR) and the serious games project by Marfisi-Schottman et al (2010).

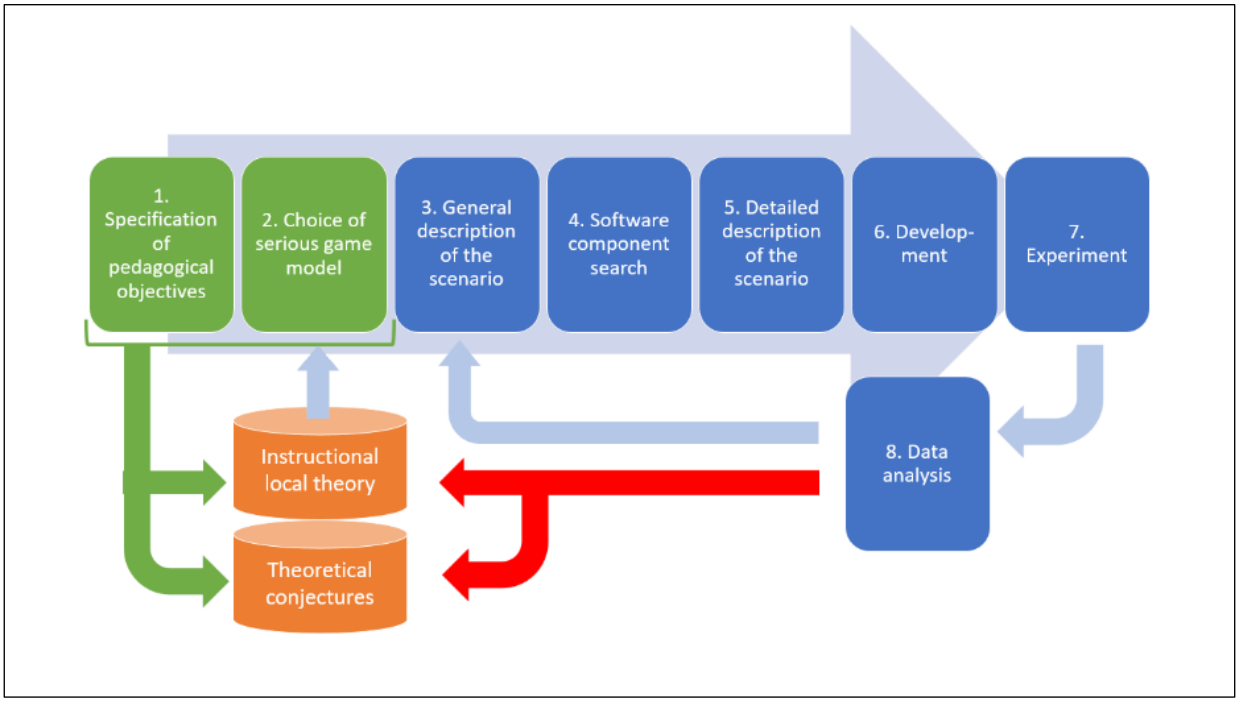

Figure 1: Model of development and research process.

Design-Based Research (DBR) is characterized by an iterative and interventionist (Brown, 1992; Gravemeijer & Cobb, 2006) process in which the goal is to achieve useful, practical and reproducible educational artifacts and learning theories in the real world (Cocciolo, 2005; Choi et al., 2017). The first iterations usually present few and fragile results. However, the iterative nature of the methodology allows the theories to evolve, be adjusted and optimized during interventions (DBRC, 2003; Akker et al., 2006). Researchers experiment with theories through prototypes until they mature their ideas into a more robust theory. The evolution of the prototype, and consequently of the underlying theories, contributes to the understanding of the actions that can lead or not to learning (Walker, 2006; Majgaard et al., 2011). This understanding happens exclusively through experiences, which produce and are produced by new theories (DBRC, 2003), and happen, above all, in a real world context (Choi et al., 2017).

In addition to the iterative and incremental cycles proposed by DBR, the model considered the serious games development process proposed by Marfisi-Schottman et al. (2010), which is suitable for minimalist teams and explores pedagogical theories applied to games in order to create the best teaching and learning conditions.

This research was developed in four cycles, each in a different semester. The experiences did not limit students either in time or space, as they were conducted considering free use of the game at any time and place, allowing us to possibly evaluate results closer to reality in terms of flow and learning. This paper presents the main findings of the last cycle, which took nine weeks at University of Santa Catarina State (Brazil) in an introductory programming course, included in the first semester of the Software Engineering bachelor's degree. The whole development process is described in Vahldick et al. (2020).

4.2 Participants

The participants were 33 undergraduate students of University of Santa Catarina State (Brazil) enrolled in an introductory programming course of the first semester of the Software Engineering bachelor's degree. The sample was composed of 87.9% of males and 12.1% of females (mean age = 22.5, SD = 4.66). 81.8% of the students reported that they had no previous programming experience. In addition, 33.3% of them declared that they play digital games every day and 24.2% of them declared that they do it seldom.

4.3 NoBug's Snack Bar: a blocks-based serious game to support programming learning

When NoBug's Snack Bar design and development process was initiated, a few decisions were made. It was designed as a web-based game inspired by time management games. The player should control the attendant of a snack bar using programming commands. Customers are controlled by the game, and they make requests that are combinations of foods and drinks. The attendant should perform the necessary steps to fulfill the requests. Each mission ends when the player fulfills a certain number of requests.

The game frontend was coded in HTML5 and Blockly (Fraser, 2015) was used for the construction of resources with blocks. It transforms blocks into Javascript code, allowing it to be executed in a browser without the need for compilation and execution on the server side, thus reducing latency by avoiding the transmission between client and server. The features of the game and its functionality are presented in detail in Vahldick et al. (2020). Figure 2 illustrates the main interface of the game. Although the game was used in Brazil and Portugal, it was developed to support a multilingual interface. Menus and buttons are shown according to the language selected. Only the blocks kept their names in English, so that students get used to them when they program in real programming languages.

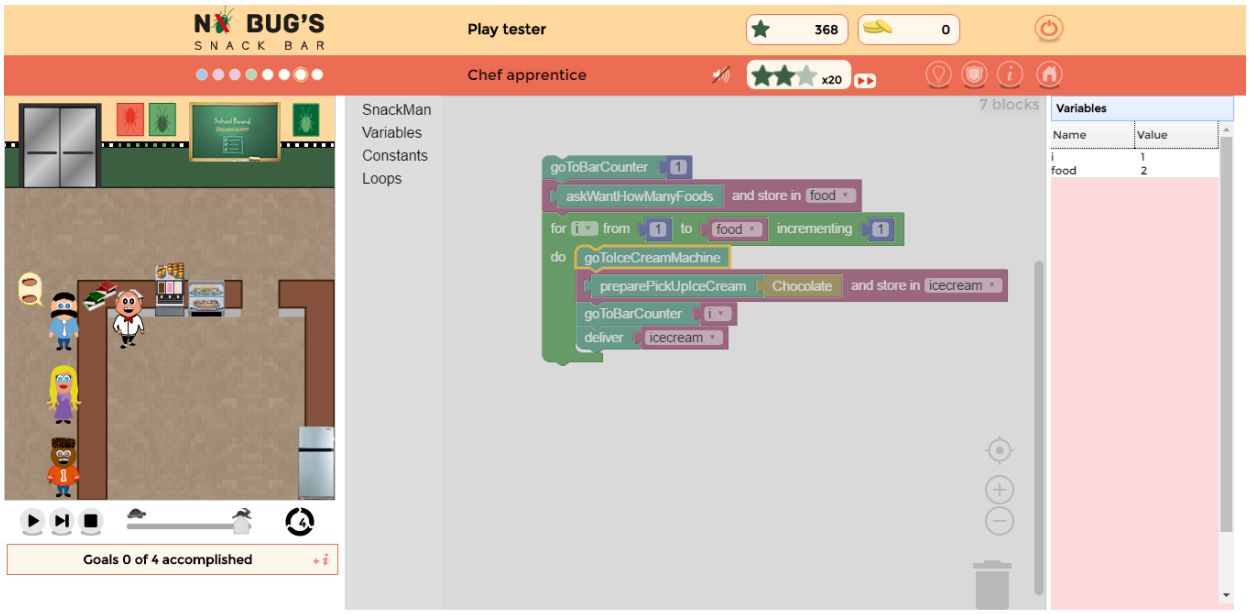

Figure 2: Main interface of the game.

The game covers the initial topics usually included in introductory programming courses. They are organized in ten levels with a total of 74 missions: levels 2 to 4: Variable manipulation (19 missions); levels 5 to 7: Conditionals (22 missions); levels 8 to 10: Loops (24 missions: for loop, while loop and the two together). The initial level, level 1 (9 missions), is an introductory level to learn to play the game. Although students had freedom to choose their next mission after

level 1, the suggested learning sequence is illustrated in Figure 3. Each circle represents a level, the arrows represent the prerequisites between levels, and within parenthesis is indicated the number of missions in each level. White levels include the essential missions. Students should learn the basic concepts at these levels. Light grey levels are enhancement levels and dark grey levels are mastering levels. Students can practice new and more complex situations in enhancement levels. Mastering levels have very challenging missions adequate to the better performing students.

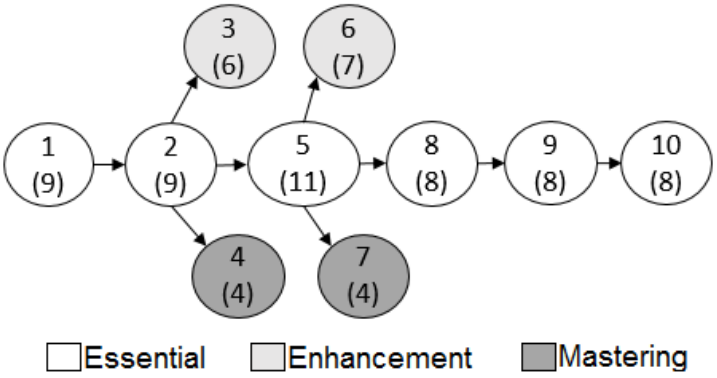

Figure 3: Main interface of the game.

In addition, within each level, the types of tasks asked in the missions were organized considering Bloom's Taxonomy of educational objectives in the cognitive domain (Anderson et al., 2001). The idea was to get a better alignment between the mission's activities and desired to learn outcomes. The more gradual increase in difficulty could also make the game more interesting to the students (Lameras et al., 2017). The first type of task is multiple-choice mission, where a solution is provided and a question about it is presented with four answer options. The student selects one of the options. The game runs the solution and verifies the student's answer. After, a solution is provided with some errors. Students must fix the errors to accomplish the mission. The types of errors can be incorrect use of comparison or logical operators, erroneous references of variables, or wrong sequence of blocks. The student must correct the mistakes by changing operators, variables or the order of the blocks. Next, all the solution blocks are provided but dispersed in the workplace. The student must sort the blocks in the right order. Then, a partial solution is provided with some blocks missing. The student must complete the solution. And finally, the student creates her/his own solution from scratch, sometimes with a started code.

4.4 Instruments and procedures

This section describes the instruments and procedures used to answer the previously mentioned four research questions. The process and instruments describe below are illustrated in Figure 4. The students were informed that in that semester they would be using a game being developed as a research resource by another teacher from University. This game would serve as content and practice for the first contact with computer programming activities. It was explained to the students that the whole experience was monitored by the other teacher, who would even serve as a support for their difficulties. Finally, the students were informed that the results and conclusions of this research could be published in journals and conferences, respecting their anonymity.

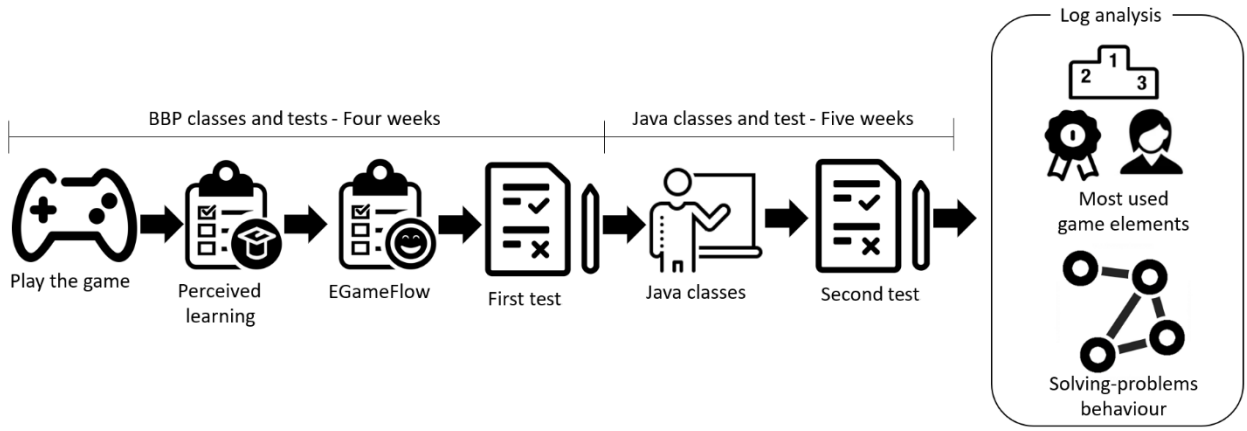

Figure 4: Instruments and process of the last cycle in the research

RQ1. What is the difference between the groups regarding the feelings about their learning? Students were encouraged to play during the first three weeks of the course. After that period, a questionnaire was applied (Table 1) using a 5 points *Likert* scale (1- totally disagree and 5-totally agree) to measure students' feelings about their learning. Perceived learning is a set of beliefs and feelings regarding the learning that has taken place and reflects the student's sense that some new knowledge has been acquired and some new understanding has been reached, even if this subjective knowledge and understanding contrasts with academic performance (Caspi & Blau, 2011). The perceived learning represents the degree of confidence that the student has regarding his/her mastery of a given topic. Although question 07 was not directly related to the evaluation of perceived learning, the opportunity was taken to ask students' opinions on the continued use of the game in classes.

Table 1. Questionnaire to measure the perceived learning.

#	Question
01	I learned from the game
02	I learned from the game as using variables
03	I learned from the game as using conditionals
04	I learned from the game as using loops
05	I learned from the game that it makes my job easier to divide the problem into smaller parts
06	I learned from the game as is important to debug to fix errors
07	I recommend using the game in the next semester

RQ2. What is the difference between the groups regarding the feelings about fun? Then students were submitted to the first test. Before the start of the test an adapted version of the EGameFlow instrument (Fu et al., 2009) was used to assess students' views about their experience playing the game. Originally there were 42 questions distributed among eight dimensions. Two dimensions have been removed: social interaction and knowledge improvement (11 questions). The social interaction dimension presented questions regarding multiplayer games, player interactions, and community formation, all issues not addressed in this research. Knowledge improvement was evaluated using the questionnaire shown in Table 1. In addition, some ambiguous questions were removed and the wording of other questions was adapted to refer to learning programming. The 26 questions used are included in Table 2.

Table 2. EGameFlow adapted.

Factor	Question	Factor	Question
Concentration	Most of the gaming activities are related to the programming task	Challenge	The difficulty of challenges increased as my skills improved.
	I am not burdened with tasks that seem unrelated		The difficulty of challenges was compatible with my knowledge of programming
	Workload in the game is adequate		The difficulty between one mission and the next are adequate
Goal Clarity	Overall game goals were presented clearly	Autonomy	I feel a sense of control over the game
	Intermediate goals were presented clearly		I understand the type of action performed by each block type.
Feedback	I receive feedback on my progress in the game	Immersion	I can become involved in the game
	I receive immediate feedback on my actions		I feel emotionally involved in the game
	I receive enough feedback to solve the missions.		I experience an altered sense of time
	I understand the error messages the game shows me		The context of the game (snack bar) is attractive
	I understand what goals I can't accomplish that the game shows me		I get unmotivated by the graphic quality of the game
	The game provides “hints” in text that help me overcome the challenges		The background music helps set the pace of my work
	The emails sent with “hints” helped me to solve the problems in the missions.		I get motivated to play for points that allow me to access avatar customization
			I get motivated to be in the best positions on the leaderboards
			The emails sent with “hints” motivated me to keep playing

The test was a conventional, classroom, paper-based test, including 5 questions: (Q1) indicate the output of a given code; (Q2) answer a multiple-choice question about a given code; (Q3) correct a given code that had two errors; (Q4) complete a code; (Q5) create a solution from scratch. The questions were common to many introductory programming tests, namely factorial calculus (Q3), Body Mass Index calculation (Q4), and number division using successive subtractions (Q5). Students did not use the NoBug’s context during the test, but they were expected to answer using the representation of the block. After this test, students had five weeks of classes using Java and were submitted again to another test. The Java lessons used the knowledge learned from the game, even getting some students to play again. This second test included five questions that asked for Java code creation. Again, none of them used any NoBug’s specific commands or context.

RQ3. What is the most used game element not directly related to learning? Three elements in the game are not directly related to learning. They aim to keep students’ motivated to continue playing to win points: achievement system, leader board and avatar customization. To assess these elements' usefulness, we considered how often the students accessed each of them. It was also interesting to verify if there was any preference between the three items within the two groups to

better explore motivation and learning in the future. The following students' interactions were considered: when the student opens the avatar customization window or the achievements points window; or when the student manipulates the leader boards guides. Analyzing the logs, it was possible to know in how many sessions each of these resources was accessed by students.

RQ4. What is the student's behavior when solving problems in the game? Lastly, we analyzed how the students used the game to solve problems. For this, five types of actions that they could do in the game were considered:

1-Explanation (EXP): when the student accessed the mission explanation.

2-Depuration (DEB): when the student decided to execute her/his solution step by step.

3-Execution (RUN): when the student decided to execute her/his solution.

4-Development (DEV): when the student modified her/his solution, adding, removing, or changing blocks.

5-Revision (REV): when the student left a mission that she/he did not finish, experimented again other missions already finished, executed, or debugged them, and finally returned to the original mission.

This information was obtained by analyzing the logs.

As mentioned before, we wanted to divide the students into two groups according to their performance. We used the results of the first test, using a grade of 7.0 (the approval threshold in the institution) as a passing score (G1 < 7.0 and G2 >= 7.0). Furthermore, we considered only students that concluded at least 50% of the creation missions in the game, because the main goal in programming learning is for students to develop their own solutions to the proposed problems. As a result, G1 had 8 students and G2 had 9 students. The second half of the course (Java classes) was considered to analyze the contribution that the game may have had in improving problem-solving with programming.

4.5 Consensus measure

This section describes how the results of the two surveys (Perceived Learning and EGameFlow adapted) were used.

One of the most used measures of central tendency for summarizing the results of questionnaires given in a *Likert* scale is commonly called "average". However, from a probabilistic/statistical point of view (i.e., considering each class of the scale as a possible event of a random variable), the most coherent terminology is that of the expected value (E(x)), in the sense of "what is expected with more and more replications of the questionnaires". This measure is considered "fair" in the sense of considering all the plurality of responses, and not just the position (like the median) or the frequency (like the mode).

The expected value (E(x)) is understood as the mean value to be obtained from a random variable x when the number of its repetitions tends to infinity. For discrete random variables (such as the *Likert* scale) it can be obtained by

$$E(x) = \sum_{i=1}^{k} P(x_i).x_i \qquad (1)$$

where k is the quantity of classes, x_i is the discrete value of the class i and $P(x_i)$ is the probability of occurrence of x_i, in this context approximated by the relative frequency of the classes. It corresponds, arithmetically, to the average of the answers weighted by their respective frequency.

It is pointed out here that the *Likert* scale is of an ordinal qualitative nature, as there is a natural order between the items in the scale (i.e., between totally disagreeing and totally agreeing). To perform algebraic operations, the scale is transformed into a discrete quantity (for instance, scoring it from 1 to 5). When this is done, it is incorrectly admitted that there is a linear relationship between the levels of agreement. This linear relationship does not exist subjectively.

In this context, Tastle et al. (2005) propose a new measure to complement the interpretation of the results obtained from questionnaires. It is the Consensus Measure (Cns(x)), calculated by:

$$Cns(x) = 1 + \sum_{i=1}^{k} P(x_i) \log_2 \left(1 - \frac{|x_i - E(x)|}{X_{max} - X_{min}}\right). \quad (2)$$

Cns(x) should be interpreted as a percentage of internal agreement of the distribution concerning E(x). As $0 \leq Cns(x) \leq 1$, it is understood that the closer to 1, the more respondents "agree" with the expected value for the question. A complementary concept to the Consensus is Dissention (Dnt(x)). The Dissention is one minus the Consensus, 1-Cns(x). Dissention is defined as a difference of opinion such that strife is caused within the group undertaking to decide.

5 Results

5.1 Classes performance evaluation

To verify the impact of the game experience in learning, we examined the Pearson correlation among the two test's grades, the number of missions the students completed in the game, and the amount of time spent to complete those missions.

The average grade of the first test was 5.4 on a 0 to 10 scale (SD=2.33) and 33% of the students (n=10) had a grade equal or higher than 7.0. Four students that finished all the missions in the game had a grade equal or higher than 8.0.

The average grade of the second test was 6.0 (SD=2.48). In this test, 48% of the students (n=12) had 7.0 or higher. Only 25 students took the second test, as 8 of the initial students dropped out of the course.

Table 3 shows the correlation of the tests' results with the game experience. Positive and significant correlations among overall variables were found. There were moderate correlations among total missions played and tests' grades, and a weak correlation between time spent playing and tests' grades. Thus, students who played more had better performance in the course tests. When considering the time, the numbers are less expressive. This can be explained by the fact that some of the best students did not need much time to solve most missions.

Table 3. Pearson's Correlation among exams and game experience.

	Test 1	**Test 2**	**Total of missions**	**Time spent**
Test 1	-	0.77**	0.666**	0.452*
Test 2		-	0.641**	0.410*
Total of missions			-	0.767*
Time spent				-

** p < 0.01 * p < 0.05

In the next results, students were organized into two groups considering only students that concluded at least 50% of the creation missions and according to the results of the first test, using 7.0 as a passing score (G1 < 7.0 and G2 >= 7.0). As a result, G1 had 8 students and G2 had 9 students.

5.2 Perceived learning (RQ1)

To measure students' feelings about their learning, a questionnaire (Table 1) was applied before the first test. So, it was possible to measure the confidence that the students had before taking the test. Table 4 shows the expected value, consensus, and dissention measures for each of the two groups in each question.

Table 4. Results of perceived learning measure.

		E(X)	Cns(X)	Dnt(X)
Q01	G1	4.00	0.67	0.33
	G2	4.67	0.83	0.17
Q02	G1	4.38	0.82	0.18
	G2	4.67	0.83	0.17
Q03	G1	4.38	0.82	0.18
	G2	4.78	0.87	0.13
Q04	G1	3.60	0.70	0.30
	G2	4.88	0.92	0.08
Q05	G1	4.13	0.82	0.18
	G2	4.22	0.79	0.21
Q06	G1	4.25	0.77	0.23
	G2	4.56	0.81	0.19
Q07	G1	3.13	0.61	0.39
	G2	4.44	0.67	0.33

Analyzing the results for each question it is possible to say that there was consensus (the qualified majority (3/5) of students, that is, over 60%) in most of them. Group 1 was not confident to recommend the game in the next semester (E(Q07)=3.13). They strongly agreed that they learned to use variables and conditionals (E(Q02)=E(Q03)=4.38). The weaker students could not see the utility of the game. On the other hand, in group 2 (Cns(Q04)=0.92) students strongly agreed (E(Q04)=4.88) that the game helped them to understand loops. They also strongly agreed (E(Q03)=4.78) learned to use conditionals and learned to debug (E(Q06)=4.56) In general, it was possible to observe that G2 students had a high level of perceived learning.

5.3 Perceived fun (RQ2)

To assess the level of students' satisfaction with the game experience, they answered 26 questions that are part of the EGameFlow adapted instrument (Fu et al., 2009). Non-parametric Mann-Whitney tests were performed to assess in which items there were significant differences between the groups. Only one item showed differences with p=0.011: "Overall game goals were presented clearly". Analyzing the Mean Rank it was possible to conclude that G2 agreed more than G1 with this statement. It can be concluded that regardless of academic performance, students had the same perception about their fun, namely, the average EGameFlow score for G1 was 3.55 and for G2 was 3.64 on a scale from 1 to 5.

Originally EGameFlow was a 7-point Likert scale. According to Krosnick & Presser (2010), "Some studies have found the number of scale points to be unrelated to cross-sectional

reliability.”. Revilla et al. (2014) also point out that people can interpret the meaning of each category in different ways, and when the number of options increases, the possibility of different interpretations also increases. They concluded to offer 5 answer categories rather than 7 or 11 because of the latter yield data of lower quality. To simplify the students' interpretation, we adapted the instrument to a 5-point Likert scale.

5.4 Measuring the fun of game elements (RQ3)

Table 5 shows the number of times students in each group had authenticated, accessed the avatar customization window, leader boards guides, and the achievements points window.

Table 5. Elements of game measured.

Groups	Number of Authentications	Avatar	Leader board	Achievements
1	383	52 (13,6%)	61 (15,9%)	26 (6,8%)
2	475	48 (10,1%)	130 (27,4%)	56 (11,8%)
p-value	-	0,115	0,000*	0,013*

*p-value < 0.05

To verify if there were significant differences between the two groups, chi-square tests were performed to check if the proportions were the same between the two groups. The proportion of each item was calculated concerning the number of authentications. The most used feature by G2 students was the visualization of the leader boards, followed by the view of the achievements and the customization of the avatar. In G1, the leader boards were also the most used, followed by the avatar customization and achievements. Both groups had the leader boards as the most used element. However, G2 had a higher level of use than G1. This may indicate the intrinsic motivation of G2 students to stay in better positions, and for this they need to perform better in the game.

5.5 Problem-solving behaviors (RQ4)

The previous subsections presented statistical analyses comparing perceptions, preferences and use of game features between two groups defined based on students’ academic performance. This section aims to analyze the sequence of actions that each group of students performed within the game.

Table 6 shows the number of times the transition between one action (From) and another (To) happened in each group. Chi-square tests were performed to check the proportions between the two groups. A particular behavior of a group is considered when the proportions are different (p-value < 0.05). Based on the transitions that presented significant differences, a graph was developed for each group trying to identify the most common behaviors in each one (Figure 5). For example, from the EXP action going to the RUN action was identified a difference in proportion (p=0.032). Since G2 has a higher percentage than G1 (1.5% > 0.5%), we can conclude that this transition is more characteristic of G2. These behaviors represent what students did after their first failed execution in a mission.

When observing the G1 graph, it becomes evident that after a failed RUN, these students tend to modify (DEV) the solution and then use the debugging resources (DEB). This can be understood as that from the execution they already trust to know what is necessary to correct the solution (or are using a trial-and-error approach). After the change, they prefer to use the debugging features. Regarding G2, after rereading the explanation (EXP) they usually execute it (RUN), probably to try to identify the error not noticed even after reading it. After the failed run, this group prefers to use more debugging (DEB), indicating higher care in identifying the error(s),

instead of going right away to change the solution. While they are developing (DEV) the solution, these students still use previous solutions (REV), trying it many times. It is interesting to verify the difference between G1 and G2 after modifying the solution: G1 prefers to use more debugging, and G2 prefers more execution. In future work, the monitoring environment can incorporate the identification of these patterns so that the teacher can intervene suggesting to the student the adoption of adequate learning behaviors.

Table 6. Transitions between two actions.

From	To	G1	%	G2	%	Total	p-value
EXP	DEV	719	78.2	788	76.1	**1507**	0.253
	RUN[b]	5	0.5	16	1.5	**21**	0.032
	DEB	22	2.4	36	3.5	**58**	0.160
	REV[a]	3	0.3	3	0.3	**6**	-
	EXP	170	18.5	193	18.6	**363**	0.941
	Total	**919**	**47.0**	**1036**	**53.0**	**1955**	
DEB	DEV	432	72.2	501	72.8	**933**	0.816
	RUN[a]	1	0.2	4	0.6	**5**	-
	DEB	67	11.2	77	11.2	**144**	0.994
	REV[a]	1	0.2	5	0.7	**6**	-
	EXP	97	16.2	101	14.7	**198**	0.445
	Total	**598**	**46.5**	**688**	**53.5**	**1286**	
DEV	DEV	432	72.2	501	72.8	**933**	0.816
	RUN[a]	1	0.2	4	0.6	**5**	-
	DEB	67	11.2	77	11.2	**144**	0.994
	REV[a]	1	0.2	5	0.7	**6**	-
	EXP	97	16.2	101	14.7	**198**	0.445
	Total	**598**	**46.5**	**688**	**53.5**	**1286**	
REV	DEV	432	72.2	501	72.8	**933**	0.816
	RUN[a]	1	0.2	4	0.6	**5**	-
	DEB	67	11.2	77	11.2	**144**	0.994
	REV[a]	1	0.2	5	0.7	**6**	-
	EXP	97	16.2	101	14.7	**198**	0.445
	Total	**598**	**46.5**	**688**	**53.5**	**1286**	
RUN	EXP	46	22.8	51	20.3	**97**	0.527
	RUN[a]	1	0.05	4	1.6	**5**	-
	DEB[b]	8	4.0	36	14.3	**44**	0.000
	DEV[b]	147	72.8	160	63.7	**307**	0.041
	REV	0	0.0	0	0.0	**0**	-
	Total	**202**	**44.6**	**251**	**55.4**	**453**	

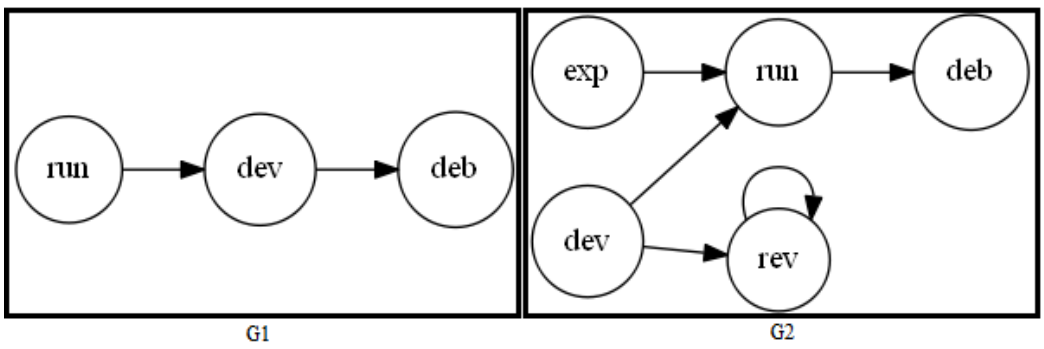

Figure 5. Students' problem-solving behaviors.

6 Limitations of the study

This study has several limitations that could pose threats to the validity of our results.

First, our analysis considers only one university in a specific course. Although students study under similar conditions, the results may not be generalized to all students due to their characteristics, for example, the school they came from. As it is common in this type of study, some variables could not be fully controlled, such as the background and previous knowledge of students.

One important limitation is the small scale of the study, 17 students in the main findings. However, capturing and analyzing the experiences of seventeen students through four weeks, allowed us to understand how and why they play the game. In addition, by collecting repeated interactions the limitation of the small sample size was minimized.

Another limitation of the study is that it only used questionnaires to collect students' opinions. Possibly other instruments could have allowed the collection of more rich information about students' feelings and opinions. Furthermore, the eight students who dropped out were not asked about their impressions of the game.

7 Discussion and conclusions

This paper presented a study about the use of a blocks-based game to support the learning of programming skills. The students were not limited either in time or space to play, as they could use the game at any time and place, allowing them to evaluate the results closer to reality in terms of fun and learning.

Moderate positive correlations were found between students' scores in the tests and the number of missions completed in the game. In the future, this result has to be further explored and confirmed, eventually leading to the development of a prediction model, to alert both the students and the teacher about situations that may lead to failure or dropout in the courses.

Below are answered the four research questions that lead this study.

RQ1. What is the difference between the groups regarding the feelings about their learning?

Overall, students with passing grades in the course (>= 7.0) felt more confident in learning with the game. This group strongly agreed that the game helped them to understand loops. However, the other group of students (grade < 7.0) agreed that the game helped them to understand variables manipulation and conditionals and was not so useful for learning loops.

RQ2. What is the difference between the groups regarding the feelings about fun?

The perceived fun was similar regardless of the performance in the course. Only one item presented some differences among the groups. G1, students agreed less than G2 students that the game explains clearly the tasks. This feeling was possibly caused by the difficulty these students felt to fulfil the missions. The average EGameFlow score was identical (G1=3.55 and G2=3.64) on a scale from 1 to 5. This scale serves as a comparison between games or groups of players in the same game.

RQ3. What is the most used game element not directly related to learning?

Some game features intended to motivate students to play the game, such as customizing the avatar, three leader boards, and an achievement system. Also, one gameplay element that has been tested, and which is not often considered in the literature about this type of game, was point

winning. We found that most students enjoyed accessing the leader board. It can be concluded that (1) even though scoring is not a primary goal of a serious game, its use can increase the attractiveness of the game, and (2) adding a leader board in serious games is interesting to most students. Serious game designers should consider including points and leader boards in their projects.

RQ4. What is the student's behavior when solving problems in the game?

It was possible to identify a difference in the two groups' behavior when solving the missions. This may indicate the same behavior when solving programming exercises without using the game: the top-ranked students often accessed the solutions of previous missions to help them solve a new mission, while the other students usually used a trial-and-error approach. In future work, the game can evaluate this behavior and suggest that the student access a specific previous activity, or even highlight part of the text of the statement that should be considered more carefully to solve the mission.

Although we did not find block-based games applied to programming learning for undergraduate computer science courses, the related work served to compare features in NoBug's. Firstly, despite NoBug's does not apply the Parson Problems where students put code fragments together to construct a program, there is one type of mission where lines of code are missing. So, before the student reaches this type of assignment, and the following ones that are for creating programs from scratch, she/he learns the concepts through examples, with assignments to solve errors and put blocks in order. Crescendo provides the problems as challenges for the students to solve, in a Use-Modify-Create sequence, very similar to the sequence of tasks in NoBug's, in which initially the student observes, then modifies, and finally creates from scratch. Students learn to use the programming structures independent of the language in NoBug's. On the other hand, BlockPy is well integrated BBP with Python which can make it easier for students to learn when they can see both representations simultaneously. It can even increase their belief in the usefulness of the game and see a greater relation to the subjects studied in lectures. This idea can inspire us to provide the same approach as future work.

All procedures performed in this study were following the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.

Acknowledgments

First author acknowledges the doctoral scholarship supported by CNPq/CAPES – Programa Ciência sem Fronteiras – CsF (6392-13-0) and authorized retirement by UDESC (688/13). We also want to thank the students that played the game and their teachers that allowed us to try it with them.

References

Akker, J. van den, Gravemeijer, K., McKenney, S., & Nieveen, N. (2006). Introducing Educational Design Research. In J. van den Akker, K. Gravemeijer, S. McKenney, & N. Nieveen (Eds.), *Educational Design Research* (pp. 3–7). Routledge. [GS Search]

Anderson, L. W., Krathwohl, D. R., & Bloom, B. S. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Allyn & Bacon. [GS Search]

Arnab, S., Freitas, S. de, Bellotti, F., Lim, T., Louchart, S., Suttie, N., … Gloria, A. De. (2012). *Pedagogy-driven design of Serious Games: An overall view on learning and game mechanics mapping, and cognition-based models*. *Research Report*. DOI: 10.1111/bjet.12113 [GS Search]

Bart, A. C., Tibau, J., Kafura, D., Shaffer, C. A., & Tilevich, E. (2020). Design and Evaluation of a Block-based Environment with a Data Science Context. *IEEE Transactions on Emerging Topics in Computing*, *8*(1), 182–192. DOI: 10.1109/TETC.2017.2729585 [GS Search]

Ben-Ari, M. (2013). Visualization of programming. In *Improving computer science education* (pp. 52–65). [GS Search]

Boller, S., & Kapp, K. (2017). *Play to Learn: Everything You Need to Know About Designing Effective Learning Games*. ATD Press. [GS Search]

Bosse, Y., & Gerosa, M. A. (2017). Difficulties of Programming Learning from the Point of View of Students and Instructors. *IEEE Latin America Transactions*, *15*(11), 2191–2199. DOI: 10.1109/TLA.2017.8070426 [GS Search]

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences*, *2*(2), 141–178. DOI: 10.1207/s15327809jls0202_2 [GS Search]

Caspi, A., & Blau, I. (2011). Collaboration and psychological ownership: How does the tension between the two influence perceived learning? *Social Psychology of Education*, *14*(2), 283–298. DOI: 10.1007/s11218-010-9141-z [GS Search]

Choi, J., Lee, Y., & Lee, E. (2017). Puzzle Based Algorithm Learning for Cultivating Computational Thinking. *Wireless Personal Communications*, *93*(1), 131–145. DOI: 10.1007/s11277-016-3679-9 [GS Search]

Cocciolo, A. (2005). Reviewing design-based research. Retrieved February 14, 2014, from <http://www.thinkingprojects.org/wp-content/dbr.doc>.

DBRC. (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher*, *32*(1), 5–8. DOI: 10.3102/0013189X032001005 [GS Search]

Fraser, N. (2015). Ten Things We ' ve Learned from Blockly. In *IEEE Blocks and Beyond Workshop* (pp. 49–50). DOI: 10.1109/BLOCKS.2015.7369000 [GS Search]

Fu, F. L., Su, R. C., & Yu, S. C. (2009). EGameFlow: A scale to measure learners' enjoyment of e-learning games. *Computers and Education*, *52*(1), 101–112. DOI: 10.1016/j.compedu.2008.07.004 [GS Search]

Gravemeijer, K., & Cobb, P. (2006). Design research from a learning design perspective. *Educational Design Research*, 17–51. [GS Search]

Holbert, N. R., & Wilensky, U. (2011). FormulaT racing: Designing a game for kinematic exploration and computational thinking. In *7th International Conference on Games + Learning + Society*. Madison, USA. [GS Search]

Johnson, L., Becker, S. A., Estrada, V., & Freeman, A. (2015). *Horizon Report: 2015 Higher Education Edition*. *Reading*. Austin, Texas: The New Media Consortium. Retrieved October 10, 2021, from <https://files.eric.ed.gov/fulltext/ED559357.pdf>.

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2013). Understanding computational thinking before programming: Developed guidelines for the design of games to learn introductory programming through game-play. In P. Felicia (Ed.), *Developments in Current*

Game-Based Learning Design and Development. Hershey, PA: IGI Global. DOI: 10.4018/ijgbl.2011070103 [GS Search]

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming. *ACM Computing Surveys*, *37*(2), 83–137. DOI: 10.1145/1089733.1089734 [GS Search]

Koster, R. (2014). *A Theory of Fun for Game Design* (2nd ed.). O' Reilly Media, Inc. [GS Search]

Krosnick, J. A., & Presser, S. (2010). Question and Questionnaire Design. In P. V. Marsden & J. D. Wright (Eds.), *Handbook of Survey Research* (2nd ed., pp. 263–313). Bingley, UK: Emerald Publishing Limited. DOI: 10.1007/978-3-319-54395-6_53 [GS Search]

Lameras, P., Arnab, S., Dunwell, I., Stewart, C., Clarke, S., & Petridis, P. (2017). Essential features of serious games design in higher education: Linking learning attributes to game mechanics. *British Journal of Educational Technology*, *48*(4), 972–994. DOI: 10.1111/bjet.12467 [GS Search]

Laurillard, D., Charlton, P., Craft, B., Dimakopoulos, D., Ljubojevic, D., Magoulas, G., … Whittlestone, K. (2013). A constructionist learning environment for teachers to model learning designs. *Journal of Computer Assisted Learning*, *29*(1), 15–30. DOI: 10.1111/j.1365-2729.2011.00458.x [GS Search]

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., … Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32–37. DOI: 10.1145/1929887.1929902 [GS Search]

Majgaard, G., Misfeldt, M., & Nielsen, J. (2011). How design-based research and action research contribute to the development of a new design for learning. *Designs for Learning*, *4*(2), 8–27. [GS Search]

Marfisi-Schottman, I., George, S., & Tarpin-Bernard, F. (2010). Tools and Methods for Efficiently Designing Serious Games. In *4th European Conference on Game-Based Learning* (pp. 226–234). Copenhagen, Denmark. [GS Search]

Mazlan, M. N. A., & Burd, L. (2011). Does an avatar motivate? In *41th Annual Frontiers in Education Conference*. Rapid City, USA. DOI: 10.1109/FIE.2011.6142700 [GS Search]

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* New York: Basic Books, Inc. [GS Search]

Prensky, M. (2001). *Digital game-based learning*. New York: McGraw-Hill. [GS Search]

Razak, A. A., Abidin, M. I. Z., & Connolly, T. M. (2019). Transitioning to Digital Games-based Learning: The Case of Scottish Universities. In A. Visvizi, M. D. Lytras, & A. Sarirete (Eds.), *Management and Administration of Higher Education Institutions at Times of Change* (pp. 151–165). Emerald Publishing Limited. DOI: 10.1108/978-1-78973-627-420191009 [GS Search]

Revilla, M. A., Saris, W. E., & Krosnick, J. A. (2014). Choosing the Number of Categories in Agree-Disagree Scales. *Sociological Methods and Research*, *43*(1), 73–97. DOI: 10.1177/0049124113509605 [GS Search]

Robins, A. V. (2019). Novice programmers and introductory programming. In S. A. Fincher & A. V. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (pp. 327–376). Cambridge, UK: Cambridge University Press. [GS Search]

Shabalina, O., Malliarakis, C., Tomos, F., & Mozelius, P. (2017). Game-based learning for learning to program: From learning through play to learning through game development. In

Proceedings of the 11th European Conference on Games Based Learning, ECGBL 2017 (pp. 571–576). Graz, Austria. [GS Search]

Sorva, J., Karavirta, V., & Malmi, L. (2013). A Review of Generic Program Visualization Systems for Introductory Programming Education. *ACM Transactions on Computing Education*, *13*(4), 15.1-15.64. DOI: 10.1145/2490822 [GS Search]

Tastle, W. J., Russell, J., & Wiermann, M. J. (2005). A new measure to analyze student performance using the Likert scale. In *Information Systems Education Journal*. [GS Search]

Vahldick, A., Farah, P. R., Marcelino, M. J., & Mendes, A. J. (2020). A blocks-based serious game to support introductory computer programming in undergraduate education. *Computers in Human Behavior Reports*, *2*(October), 100037. DOI: 10.1016/j.chbr.2020.100037 [GS Search]

Vahldick, A., Mendes, A. J., & Marcelino, M. J. (2014). A review of games designed to improve introductory computer programming competencies. In *44th Annual Frontiers in Education Conference* (pp. 781–787). Madrid, Spain. DOI: 10.1109/FIE.2014.7044114 [GS Search]

Walker, D. (2006). Toward productive design studies. In J. van den Akker, K. Gravemeijer, S. McKenney, & N. Nieveen (Eds.), *Educational Design Research* (pp. 9–19). Routledge. [GS Search]

Wang, W., Zhi, R., Milliken, A., Lytle, N., & Price, T. W. (2020). Crescendo: Engaging students to self-paced programming practices. In *SIGCSE* (pp. 859–865). DOI: 10.1145/3328778.3366919 [GS Search]

Weintrop, D., Holbert, N. R., Wilensky, U., & Horn, M. (2012). Redefining Constructionist Video Games: Marrying Constructionism and Video Game Design. In *Constructionism 2012* (pp. 645–649). Athens, Greece. [GS Search]

Weintrop, D., & Wilensky, U. (2014). Situating programming abstractions in a constructionist video game. *Informatics in Education*, *13*(2), 307–321. [GS Search]

Weintrop, D., & Wilensky, U. (2016a). Bringing Blocks-based Programming into High School Computer Science Classrooms. In *Annual Meeting of the American Educational Research Association*. Washington, USA. [GS Search]

Weintrop, D., & Wilensky, U. (2016b). Playing by Programming: Making Gameplay a Programming Activity. *Educational Technology*, *56*(3), 36–41. [GS Search]

Xinogalos, S., Satratzemi, M., & Malliarakis, C. (2017). Microworlds, games, animations, mobile apps, puzzle editors and more: What is important for an introductory programming environment? *Education and Information Technologies*, (22), 145–176. DOI: 10.1007/s10639-015-9433-1 [GS Search]

Zhi, R., Chi, M., Barnes, T., & Price, T. W. (2019). Evaluating the effectiveness of parsons problems for Block-based programming. In *ICER 2019* (pp. 51–59). DOI: 10.1145/3291279.3339419 [GS Search]