Article

Learning Basic Programing Skills With Educational Games: A Case of Primary Schools in Slovenia

Journal of Educational Computing Research
0(0) 1–26

DOI: 10.1177/0735633116680219
jec.sagepub.com
SAGE

Franc Jakoš[1] and Domen Verber[2]

Abstract

This study aimed to investigate the effectiveness of using educational games for learning basic programing skills. For this, we designed the game "Aladdin and his flying carpet" with all pedagogical consideration in mind. We conducted the experiment with 107 sixth grade pupils with no prior knowledge of programing. The game was applied at the beginning of the course in a naturalistic classroom environment. The progress was monitored and evaluated with validated pre- and posttests. The knowledge gain was observed according to the set of learning objectives from the general curricula, and minimum knowledge standards were pursued within those. The result demonstrated that most of the pupils achieved all of the observed learning objectives. The biggest progress was observed with the "complete a program" objective. The least progress was observed with the tasks where "create a program" and "divide a problem" objectives were combined. Strong correlation was found between the final results of the posttest and the scores from playing the game, especially with the more challenging "divide a problem" learning objective. Finally, no significant difference was observed in the results between girls and boys.

Keywords

computer science education, visual programing, educational games, primary school, assessment of knowledge

[1]Janka Glazerja Ruše Primary School, Ruše, Slovenia
[2]Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia

Corresponding Author:
Franc Jakoš, Janka Glazerja Ruše Primary School, Selnica ob Dravi Primary School, Lesjakova ulica 4, Ruše 2342, Slovenia.
Email: franc.jakos@glazer.si

Introduction

Programing is a very useful skill and can be a rewarding career. In recent years, the demand for programmers has grown rapidly. For this reason, the number of students interested in the programing skill and the number of introductory programing courses have also increased. It has become clear that computational thinking has become one of the basic skills that every young person should obtain, regardless of their future profession (Fletcher & Lu, 2009). In Slovenia, the curriculum of the Computer Science Course also includes teaching the programing principles (Ministry of Education, Science and Sport, 2015). In the school year 2014/2015, it was introduced in the fourth grade as an elective course. As a consequence, only a few pupils acquire such knowledge in primary schools. The main challenge is how to attract pupils to take the CS Course and how to provide pupils access to knowledge, which would enable them to become creators and designers of technology, and not only the users (UNESCO, 2013). Therefore, we need motivational tools that would be well suited for the beginners and would provide the most effective learning experience for all pupils. One of the instruments that we can use is playing the games, which can be very effective for the novices (Lode, Franchi, & Frederiksen, 2012).

Various reports (THE A-GAMES PROJECT, University of Michigan, 2016, Wastiau, Kearney, & Berghe, 2009) on using educational games indicate that they are powerful instruments, which need to be considered. Because of that, we decided to use educational games to assist the learning of basic programing skills for the novice programmers. The first challenge was designing a proper educational game, which can be used for an introduction to programming, based on the findings of prior research and with pedagogical and contextual consideration in mind. For this purpose, we created an educational game named ''Aladdin and his flying carpet.'' Next, the appropriate application was defined for use in class. The game was introduced at the beginning of the Computer Science Course in the sixth grade of primary schools. All the participants were exposed to the same conditions of the experiment with specified playing time. We were interested in what learning outcomes can be achieved with playing the game, considering the chosen learning objectives from the curricula. That would give us a coherent base for future research in providing a complete introductory course of Computer Science in primary schools.

The second challenge was designing and validating assessments. The goal was to use programing language independent valid and reliable assessments (CSTA, 2015). For this, we prepared a series of pre- and posttests with simple tasks. The construct of a task is composed of pseudocode, which describes the travelled path of the object on the grid (Jakoš & Lokar, 2015). Validation included an expert panel review, ''think-aloud'' interviews with pupils and IRT (Item Response Theory) analysis.

After the design and validation of assessment were discussed, the procedure was described, which includes collecting the data and setting the metric to define

minimum standards of knowledge. Finally, the results are analyzed and discussed. The article concludes with limitations of the research and future work directions. We could evolve the game further and plan a new extended application for use in class, where some sort of "learning momentum" could be observed and utilized more effectively.

Literature Review

Computer Programing Learning and Teaching

Current pedagogical theory focuses on pupils' learning. Therefore, it is important to address the kinds of mental models pupils have and how those models change (Winslow, 1996). Effective communication between teacher and pupils is also important. Robins, Rountree, and Rountree (2002) found from a survey in an Introductory Programing Course, that the most reliable predictor of students' success was the expected grade. Most of the important factors lie in motivation, confidence, and positive emotional responses. Any differences that exist among pupils can be overcome with finding mutual goals (Sedighian & Sedighian, 1996). Such goals can later be achieved by using attractive learning tools. Appealing environments need to be built with different design and pedagogical consideration in mind. Soloway and Spohrer (1989) outline several suggestions relating the design of learning environments used for teaching novices: the use of graphical languages with visible control flow, simple, and consistent naming conventions, graphical animation of program states, and gradual withdrawal of initial support and restrictions.

Considerable promise has been shown with the use of visual programing languages, where programing is done with graphical code building blocks, such as Scratch (MIT Media Lab, 2016). Studies where Scratch was used for introduction to programing have shown very good results (Meerbaum-Salant, Armoni, & Ben-Ari, 2010). The same authors have found that students included in these studies gained some habits that are not recommended for further education in programing. These are bottom-up development process starting with the most basic commands and a tendency to extremely fine-grained programing. However, the ability they have, to construct quickly even more complex programs, is more important (Meerbaum-Salant, Armoni, & Ben-Ari, 2011). One of the key elements of visual programing tools is its instant feedback and inability to create a syntax error. A study by Cunniff and Taylor (1987) compared the speed and correctness of responses given by novice programmers who used visual programing languages, and those who used textual programing languages. On almost all the tests, the students who used visual programing languages performed better. In comparison, between the visual programing languages, Scratch, Alice (2015) and Greenfoot (Programming Education Tools Group, 2015), Scratch is found to be the most suitable for beginners (Ward & Bell, 2009).

Using Games to Facilitate Learning

The traditional educational dogma supports the concept of "stealth learning" where students are encouraged to have fun and learn. Students think they are merely playing, but they are learning simultaneously (Sharp, 2012). Teachers introduce learning goals through non-traditional tools, such as games. Compared with the formative assessment approach, where pupils and teachers create common learning goals, the stealth learning approach appears to hide its learning goals. Papert (1998) argues that an approach with concealed learning purpose is immoral and can hinder the learning experience. According to Dijkstra (2016), graphical programing environments are misleading and represent an unacceptable "dumbing down" of the process. However, a more relaxed and familiar learning procedure is often used to shorten the initial learning curve.

Almost all pupils play digital games or are also engaged in social networks. They frequently participate in activities where they are rewarded for their dedication and effort. Consequently, in order to ensure successful learning outcomes, educators have started to search for mutual learning goals with using immersive environments and gaming technologies (Felicia, 2009). The aim is that, while playing games, the players engage to such an extent that they learn the programming concepts stealthily (Lode et al., 2012). This is based on the premise that the best learning is interactive learning, where each succeeding exercise is a bit more difficult than pupils' current level of knowledge (Norving, 2016). The constructivist approach is often used to ensure that learning objectives are achieved. Such approach prioritizes learning from experience or learning by doing, where pupils learn by interacting with their environment and peers (Lewis & Williams, 1994). Pupils can also observe other peers and collaborate to improve their skills. This involves a process of trial and error and the learners' ability to interpret their past and present experience to update their knowledge (Felicia, 2009). Suitably designed educational computer games may increase intrinsic motivation through challenges, curiosity, and imagination (Maragos & Grigoriadou, 2007). They can provide practical examples of concepts and rules that would otherwise be difficult to illustrate in the real world. In addition, some pupils might prefer experimenting and learning from their mistakes.

Creators of computer games and educators have joined forces and created "serious games" for education, where learning is more important than entertainment (Freitas, 2007). Egenfeldt-Nielsen (2007) comes to the conclusion that it is not easy to combine the learning goals and fundamental structure of the game into an interesting plot, in a way that it would keep students intrinsically motivated to progress in the game. For example, it is easy to lose focus on the pursued educational goals by putting too much emphasis on the appearance of the game. Similarly, we may achieve learning objectives but neglect the gaming features. Consequently, such games have been classified as edutainment (Egenfeldt-Nielsen, 2015). There are very few studies that would suggest

which features of games impact learning outcomes the most (Wilson et al., 2009). Otherwise, the game makers could build game elements with the certain learning goals, and the teachers could select the appropriate game for achieving the objectives. Nevertheless, nowadays, it is easier for educators to develop games thanks to available game engines. They enable them to focus more on educational features and less on technology. Recently, virtual environments such as Second Life (Linden Research, 2016) and Opensimulator (2016), have been used to teach programing skills (Rico, Martínez-Muñoz, Alaman, Camacho, & Pulido, 2011). Furthermore, these immersive features have been coupled with Moodle (2016), which enables educators to monitor students' progress.

In terms of the learning process, digital games can be analyzed through models such as Carroll's minimalist theory (1998), Vygotsky's Zone of Proximal Development (1978) or Kolb's basic learning model (Kolb & Fry, 1975). To some extent, the cycle of learning events in digital games can be compared with Kolb's learning cycle. First, they experience failure and then they identify the cause of the failure. Next, they plan the actions that might overcome the problem and test their new plan.

The observations in the European Schoolnet Report (Wastiau et al., 2009) indicate that the use of games in the classroom has some benefits. In practice, teachers observe renewed motivation and progress of certain skills (social, intellectual, spatio-temporal, etc.). Digital games that include elements of formative assessment are also studied in the Report (THE A-GAMES PROJECT, University of Michigan, 2016). Teachers who use games for formative assessment conduct assessment more frequently and report fewer barriers.

Principles of designing the educational game. Design of the educational game should be based on the concept of "stealth learning" and the theory of constructivism. The learning content should be wrapped carefully into the game so that the need of playing is not outshone by the feeling of practicing some abstract and irrelevant skill (Lode et al., 2012).

Several authors have identified design principles for creating an appropriate educational game. Players have to connect with the character they are playing. Being able to identify with it and having control over its actions creates a strong feeling of agency (Gee, 2005b). Intrinsic motivation is best achieved when clear goals are set with some relevance for the player (Sedighian & Sedighian, 1996). Flow theory can be considered as a universal model of enjoyment. It requires from game designers to use challenges that are matched with a player's skill level. Also, the game must contain a frame story with clear goals, immediate and unambiguous feedback, a sense of control, playability (Kiili, 2005) (Gee, 2008). A narrative or fantasy principle provides a mission problem where content and context have to be balanced carefully in order not to distract pupils from the learning content (Honey & Hilton, 2011). A young audience is especially less

forgiving to the shortcomings of a game lacking visual stimuli. Nevertheless, we have to be careful in employing visual and auditory stimuli together. According to Zagami (2008), programing languages that are predominantly visual or auditory (but not both) reduce cognitive load. Mixing visual and auditory stimuli can have a negative effect depending on the complexity of the learning concept and increase of cognitive load. Nowadays, formative assessment procedures that teachers conduct during the learning process are considered as one of the most important classroom practices to support student learning. Typically, it also involves creating goals together with pupils, making them clear and relevant for them. Assessment of knowledge is drawn from quantitate feedback, rather than final scores. Finally, some authors suggest that the game should enable players to participate in big communities, where players can share strategies, exchange files, collaborate, compete, advise each other, and o forth and, thereby, develop shared values (Shaffer, Squire, Halverson, & Gee, 2005).

Learners should be assisted with scaffoldings that are progressively, as pupils skills grow, less helpful (Gee J, 2005a). This principle offers a short learning curve, so that players can familiarize themselves gradually with the game's mechanics and become proficient. This also plays an important role in motivation. A player's motivation can be triggered by many different factors and might differ across players. Some player value the graphic and the interface of the game, some value exploration, some value more complex strategic games, some will enjoy in more simple linear scenarios that require only a short time to play to succeed. Therefore, motivation in not controlled by the reward offered for completing the task successfully, but it depends on the player's personality (Felicia, 2009). Probably no educational game can be appropriate for all players. We need to use different types of games and different learning approaches to be successful in a classroom.

Existing digital games for teaching programming. Several games that could be used for learning programing exist. Some of them require prior knowledge of specific programing languages, which may have a negative impact on the beginners' motivation. Computer games such as Robot Battle - Wikipedia (2016), Robocode Home (2016), CeeBot (2016), and COLOBOT (2016) require players to have knowledge of a particular programing language to control their robots. There are also games such as CODEHERO (Peake, 2016), CodeCombat (2016), CodeMonkey (2016), and CodeHunt (Microsoft Research, 2016), where players must enter pieces of code or change the provided code. The series of Blockly games do not require any prior knowledge of programing language. Players create programs using visual programing editors, built with the open-source Blockly library (Google Project, 2016). Its existence has made the creation of educational games much easier. For example, Blockly library is used in CodeSpell (MULTI-DIMENSIONAL, 2016), The Frozen (2016b), The Flappy Code (2016a), and so forth. The same "no-prior-knowledge" rule was applied

when creating a game called Run Marco (Allcancode, Inc., 2016). The game uses its own built-in visual programing editor and has been translated into twenty different languages. RoboZZle (Ostrovsky, 2016) and Lightbot (Niato, 2016) are games, where players try to reach the goal, that is, solve the problem and progress through levels of difficulty, by adding commands with drag-and-drop user interface. For example, in the plot of the Lighbot game, the players have to steer the robot through a certain path and switch on the lights. By limiting the number of commands, the game encourages them to use the functions for repeated actions.

RoboZZle and Lightbot also introduce conditionals and recursion to the players. The conditionals are implemented with colors. For example, in Lightbot, the command is executed if it is of the same color as the field on which the robot is currently located. For the introduction of recursion, it is also possible to call a function within a function. In our opinion, these two games are the most suitable for learning basic programing skills. Therefore, some of their principles were included in our solution.

Assessments of Knowledge for Computer Science Education

A study (Computer Science Teachers Association [CSTA; 2015]) released by the CSTA finds that there is a dearth of valid and reliable assessments for measuring student learning in Computer Science education. CSTA recommends that valid and reliable formative and summative assessments are developed for programing languages beyond Java, such as Python, C#, and so forth. Recommendation suggests the creation of language-independent assessments, with learning objectives in mind, which can also be used by researchers for comparing different learning methods and environments at the national or international level (Goldman et al., 2010).

For undergraduate students learning introductory programing with Python, MATLAB, and Java, Elliott Tew (2010) has designed the most validated First Computer Science Assessment (FCS1). She developed a multistep process for validating her exam including the following: Expert panel review, ''think-aloud'' interviews with learners from the target population, IRT analysis and observing the correlation between results from the FCS1, and results from the final exam. Miranda Parker continued Elliot's work and developed the Second CS1 Assessment (SCS1) as an isomorphic version of the FCS1. She replicated FCS1 in order to release it as the instrument for use by a broader research community (Parker, Guzdial, & Engleman, 2016).

There are a growing number of projects working toward developing assessments for the blocks-based approach (Weintrop & Wilensky, 2015). A stronger claim for their validity can be made if scores of these assessments are correlated with SCS1. We created our own version of language-independent assessments (Jakoš & Lokar, 2015).

Research Purpose

Most related studies demonstrate that the educational games are beneficial for learning in general and learning programing in particular. Several researches have been done on how games affect a player's deeper engagement and motivation. However, a few studies suggest which features of games impact learning outcomes the most. The review of literature also revealed that a little is known about the relationship between applying the educational game as an introduction to programing, at the beginning of the course to sixth grade pupils with no programing background, and the results of an assessment, which was designed with learning objectives from the curricula in mind. Moreover, most studies on teaching programing to pupils were conducted in a laboratory-learning environment and tested groups often have little diversity.

To address the abovementioned limitations, we investigated whether our proposed application of an educational game is useful for computer programing language learning and can be supported by "Aladdin and his flying carpet." The goal of primary education is that minimal standards of knowledge have to be reached by all pupils. It would be unrealistic to expect that all the pupils will meet the minimum requirements only with one learning session. Nevertheless, our hypothesis was set high and the goal was that at least 95% of the pupils will meet those standards. In addition, we were interested in any possible correlation between participants' progress and their predetermined factors, such as gender, assessment marks in science subjects and the amount of time they spent playing entertainment games.

The following research questions were addressed in this study.

-Do pupils playing the game acquire minimal standards of knowledge in relation to the observed learning goals?
-Do pupils predetermined factors have any impact on the final result?

Method

Participants

A total of 107 sixth grade pupils from three different primary schools participated in this study. Seven of them were absent in one of the phases carried out during the study, so we ruled out their results from the final analysis. There were 59 boys and 39 girls. Because all the pupils who attended the class were included in the study, we did not consider them volunteers. Pupils were divided into nine groups ranging from 10 to 15 participants. According to primary school curricula, all the participants did not take any Computer Programing Course before the experiment and were, therefore, considered as novice programmers. There was no control group. All groups were administered with the same experimental learning activity and the same pre- and posttests.

Materials

Educational game Aladdin and his flying carpet. Within the framework of our study, we have designed an educational game "Aladdin and His Flying Carpet," which can be used as an introduction to learning programing. The aim was to build a game that would support our proposed application of learning activity. We chose to use programing concepts with special emphasis on loops and subprograms as inspiration for our game-design. The intention of the game is not only to drill-and-practice but also to gain new knowledge stealthily as the learner's progress through the game.

The game was built in the multiuser virtual world simulator Opensimulator (2016). The benefits of learning programing in virtual worlds are the possibility of collaboration while creating programs in shared objects, and fast and visually rich feedback. Cloud services provided by The Academic and Research Network of Slovenia Institute (ARNES, 2015) were utilized to host our game. Moving the main character and interacting with a game environment is similar to any other MMORPG (Massively Multiplayer Online Role-Playing Game) game. The players can customize their avatar, save, and import and sell objects, import images, and alter the environment.

The plot of the game revolves around the story of Aladdin and his flying carpet. Aladdin is a fictional character, which first appeared in one of the Middle Eastern folk tales, in The Book of the Thousand Nights and a Night (Aladdin, Wikipedia, 2016). In 1992, Walt Disney Pictures released an animated feature movie with Aladdin as the main character. The movie made the Aladdin's character very attractive, especially for young pupils. Our educational game borrows the character of Aladdin, and their Robin Hood way of stealing, as depicted by the movie.

The main objective of the game is collecting the gold, which is usually positioned at the top of the tower. Aladdin's main mode of transport is the flying carpet. The pupils must develop a path for the flying carpet from the start position to the gold using a limited number of moves. On the way, several obstacles (like fire, doors, and bridges) may block the moves or end the game prematurely. The game consists of four levels with progressing complexity and an increasing amount of gold. Each level occupies a certain area in the virtual world and consists of several task sites. The pupils play the game by moving from one task site to another. When all tasks in the current level are completed successfully, the player can continue on to the next level. The game remembers what tasks were already solved and the players can resume the game at later stage. However, they cannot change the level of difficulty of a particular task.

The flying carpet flies on a route, which is specified by the program, created by pupils on the control panel. Individual commands are presented with icons. Figure 1 shows an example of a simple control panel. It contains only the basic movement commands. The program is assembled on the blank spaces below the red line. Figure 2 shows a more advanced example, with two control panels.

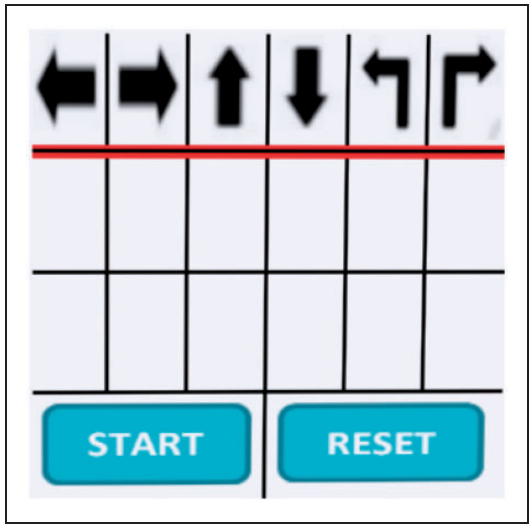

Figure 1. Simple control panel.

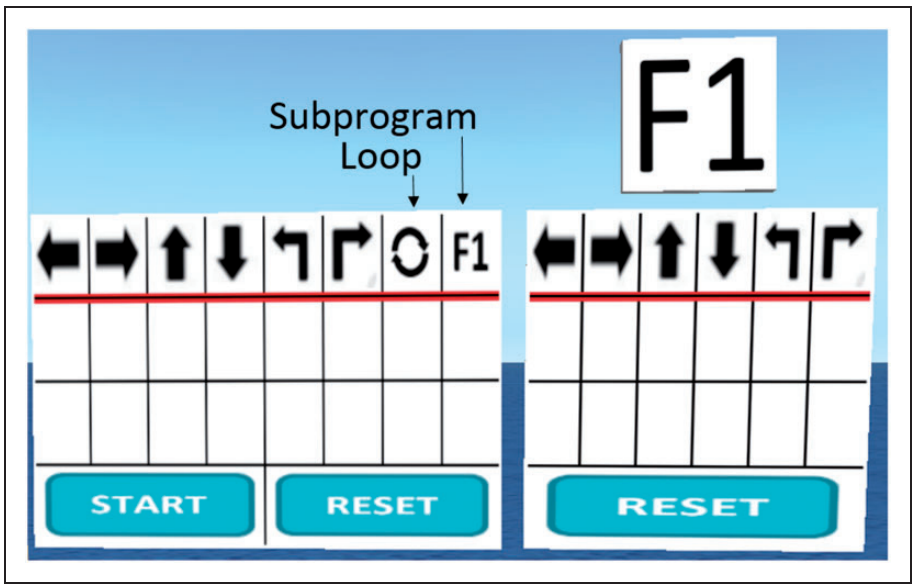

Figure 2. Advanced control panel with new icons for Loop, Subprogram call, and additional board for subprogram.

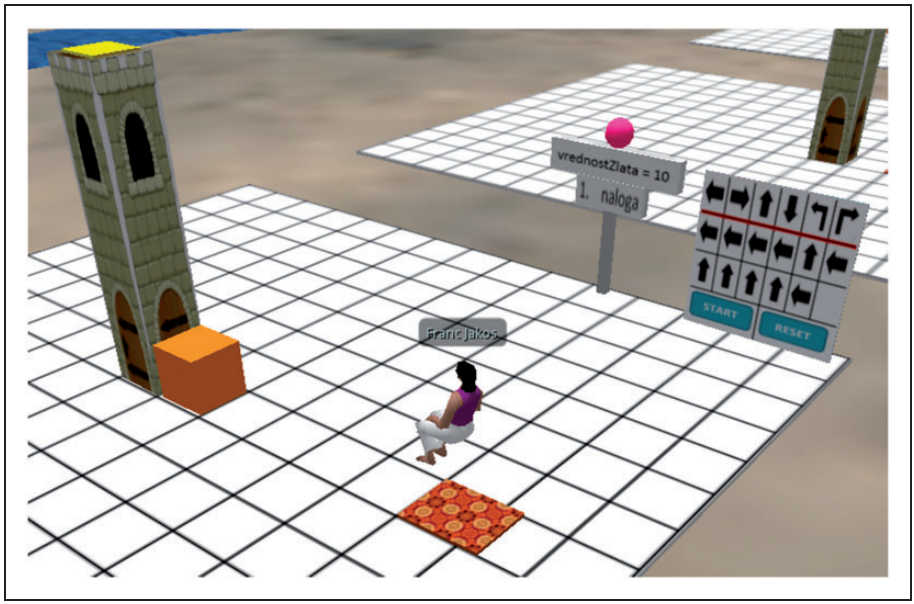

Figure 3. Constructed program or created route for carpet on simple control panel.

The first one is used for the main program and the second one for the subprogram. The main panel also includes the commands for more advanced programing concepts, such as a loop and a subprogram call. Loop performs a selected number of iterations of the next command in the program, which can also be a subprogram call.

When the program starts, Aladdin sits on the carpet and the carpet follows the instructions. At the end of the path, the pupil must steer the avatar to stand up and reach the gold. The example of the initial position is shown in Figure 3, and the end of task is illustrated in Figure 4. We must also point out that, all original words in the game are in the Slovene language.

We can categorize Aladdin according to the taxonomy of digital games (Felicia, 2009). In this game, players impersonate a fictional character through avatars. A large number of players can interact in the online virtual world. They can move through an environment and progress to the next levels of difficulty. They have to finish different tasks by solving puzzles successfully to progress further in the game. Therefore, the game falls into several different genres at once (MMORPG, Platformers, and Puzzles). We can also categorize Aladdin according to the Egenfeldt-Nielsen (2007) model of the three generations of educational computer games. In the third generation of games, which are based on the constructivist approach, meaning the player creates knowledge through experience and interaction with social communities.

Figure 4. Aladdin collects gold on the top of the tower successfully.

The goal was to build a game with all the contextual and pedagogical concerns in mind. We believe that the game is suitable for pupils from the sixth to ninth grades of primary schools. The symbols and language used in the game are adequate for this age-group (Zagami, 2008). Because the game and the players are in a multiuser virtual world, some actions made by pupils and their avatars must be prevented with additional rules. For example, avatars can be used to push other avatars, inappropriate language can be used, features meant for collaboration can be used for teasing, and so forth. The game includes

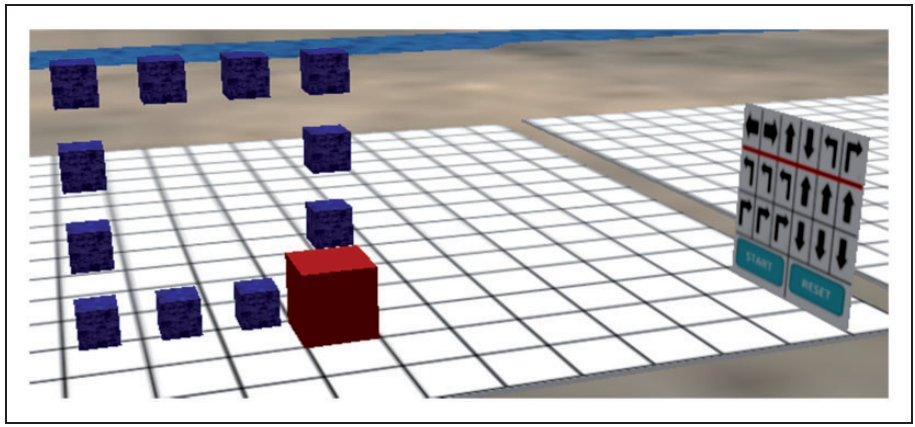

Figure 5. Sandbox area.

a scaffolding area (called a sandbox area) where pupils can learn to associate symbols on control boards with instructions for moving the carpet. For this, carpets in the sandbox area leave a trail while traveling, as is shown in Figure 5. Pupils encounter sandbox areas before they tackle their first task and the outcome does not affect the final score.

The game uses a short learning curve, allowing players to make mistakes. Only the correctly solved tasks contribute to the final score. At the beginning of the game, the tasks are simple, with a short path and a few obstacles on the way to the gold. The level of difficulty raises gradually to the point, where there is a long path with many obstacles. The game content was designed to support the basic programing concepts of loops and subprograms. The progression of the player is clear but not displayed at all times. The pupils can check only their own scores to protect players from unwanted effects that might arise from "being left behind." The teacher can choose to show all scores to players at any time. The game simulation environment supports collaboration among participants with textual chat and voice chat communication features. Players can also form groups and create programs together, or they can simply observe others while they tackle the tasks. Sloodle (2016) software was integrated into Learning Management System Moodle (2016) supports assessment of pupils' progress. Debriefing sessions were conducted at each advancement to a new difficulty level. With debriefings teachers can help pupils make a link between skills learned in the game and the learning objectives from the curricula. Virtual boards were created with questions prepared in advance. Questions were taken from reflection stems (Discover Sensors, 2016) prepared for students when reflecting on their learning. This helped pupils to look back on their experience with the game and look forward to the possible applications of their newly acquired skills.

Pupils responded to those questions by typing answers in a chat room. Certain Sloodle modules were used to link chat functionalities in the game and LMS (Learning Management System), which enabled teachers to store answers participants wrote during debriefings. This link allows implementation of the elements of formative assessment. Additional virtual boards with help were positioned at key points and were always available. However, during playing, help was used rarely and boards seemed redundant. If pupils were wrong at creating a program for a carpet, they rather just simply tried again. No print out version was provided to the pupils.

We also performed an informal usability test. The game's early version was developed almost 2 years prior to this study. During those 2 years, Aladdin was used extensively with pupils who attended ICT (Information Communication Technology) Courses. Early testing was performed with pupils from the eighth grade. To test the playability and intuitiveness of the game rules, the pupils were given no instructions on how to play the game. There were some attempts to break the game and perform undesired actions. Two main issues were identified. The first, pupils had found a way that allowed them to reach the gold without putting in the effort of creating a program for the carpet. The second, initially, there were not enough task sites to accommodate pupils playing concurrently. All issues were recorded and corrected.

We also asked the pupils who participated in testing to complete a short survey. We wanted to know if user game experience was convincing enough and did the pupils recognize our game and the virtual world as one of the entertainment games they usually play? Most of the pupils saw a strong resemblance between the virtual world experience and the entertainment games they play. However, the resemblance between our game and other games was not so convincing. There were two main reasons for that. Our game is not as sophisticated as the games they usually play, and the choice for playing them was made by teachers and not by themselves.

Assessments of knowledge. We created isomorphic copies of pre- and posttests. The only differences between them were various versions of tasks. Both tests started with the three pre-solved examples. The following tasks were divided into three different categories according to the minimal knowledge standard listed later. Pupils engaged with tasks where they had to complete, debug, or create the program. Each category contained four tasks with gradually increasing difficulty.

We have chosen to observe the next learning goals:

- The pupils acquire the ability to select the most efficient way to solve the problem.
- The pupils are able to follow instructions in the program.
- The pupils are able to identify and correct errors in the program.

- The pupils understand the concept of a loops and are able to use them to solve the problem.
- The pupils can divide the problem into sub problems.

Based on these goals, the curriculum defines minimal knowledge standards:

- The pupils complete a simple program (COP).
- The pupils create a simple program without loops (CRP).
- The pupils debug a simple program (DEP).
- The pupils divide a problem into smaller problems (DIP).

The last knowledge standard is more cognitively demanding and requires the pupil to recognize a repeating pattern in the problem. To be able to solve such tasks successfully, the pupils would have to understand the concepts of loops and subprograms.

Tests were designed according to the combination of revised Bloom and SOLO taxonomies proposed by Meerbaum-Salant et al. (2010). However, only a subset of taxonomy categories was relevant to our study. From the Bloom taxonomy, we selected the ''Applying'' and ''Creating'' categories. ''Applying'' is interpreted as the ability to read, understand, and compare the code against the object's trail on the grid. ''Creating'' is interpreted as the ability to complete, debug and, ultimately, plan and create programs based on object's trail and location in the grid. From the SOLO taxonomy, we used the ''Unistructural,'' ''Multistructural,'' and ''Relational'' categories. According to our interpretation, ''Unistructural'' means that the participants could complete, debug, or create a program with only one programing concept present. ''Multistructural'' means that they can complete, debug, or create a program with more than one programing concept present. Finally, ''Relational'' means that several programing concepts operate in relation with each other.

The basic construct of all tasks in the tests is based on a well-recognized concept used for learning basic programing skills. It consists of an object travelling on a path defined by the program. While travelling, the object leaves a trail, which makes the following of program instructions much easier. A similar approach was first used in the educational programing language Logo (Bobrow, Feurzeig, Papert, & Solomon 2016) with its turtle graphic. Each task consists of an object on a rectangular grid and a program written in a simple pseudocode. The instructions are composed of a traveling direction with the number of squares an object has to travel. An example of the task is shown in Figure 6. The red square refers to the starting point of the object. The gray squares outline the trail left behind the object. The blue square represents the object end, also the position of the object at the end of the path. Participants need to complete, debug, or create the program by observing the object's trail on the grid.

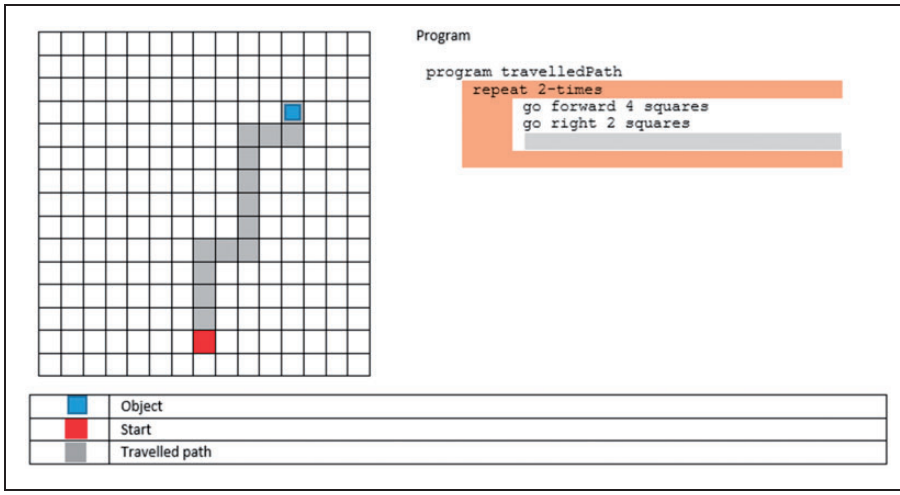

Figure 6. Task set – complete the program.

An effort was made to ensure that tasks from the game and tasks from the test have some similarities. They both use grids and programs to control object motion. The icon-based instructions in the game are slightly different from the pseudocode used in the tests. The transition would have a minor effect on the pupil's transfer of knowledge. We did not expect pupils to have problems understanding the construct of tasks, even though they tackle them for the first time. We also wanted to eliminate any prerequisites necessary for the pupils to solve the tests. For example, objects' trails are short to avoid counting mistakes. Furthermore, no coordinate system was used because this concept is introduced in the eighth grade. Instead, grids with big cells are used to define an objects location easily. In addition, we have created thorough instructions for solving the test, and every task is provided with the legend.

To be sure that assessments are measuring the desired gained knowledge, we validated tests with Elliott Taw's (2010) established methodology. First, content validity was established by expert panel review. Tasks were reviewed by experts who also participated in creating national curricula. They confirmed that each task is associated properly with the observed learning objectives. Next, think-aloud interviews were carried out with four pupils from our target population. They allowed us to investigate their reasoning while they were solving tasks. We were focused on whether the construct of tasks is understandable and if something unexpected could cause them to answer incorrectly. No significant flaws were discovered other than some typos. After the tests, we also did an IRT analysis, which was used to provide empirical evaluation of the questions' quality.

We also wanted to evaluate the correlation between the scores from existing validated tests and posttests. Miranda Parker provided SCS1, but the questions were too difficult for the target population and not in sync with the observed objectives. We were unable to find other already validated assessment suitable for our study.

Experimental Procedures

It took a lot of effort to organize experiment activities because they were all carried out based on regular schedules of different schools. We had to find courses with groups consisting of less than 15 participants and allocate them in some sort of experiment schedule. This was additionally difficult because the same teacher was assigned to all the groups. Finally, we organized experiment activities in three phases during 2 months – 2 weeks for Phases 1 and 3, and 1 month for Phase 2. During Phase 1, all the groups would have to solve pretests. The activity took one school hour (45 minutes) for each group. In Phase 2, the pupils were engaging in the game. It took approximately three school hours (135 minutes) for pupils to complete the game. The duration of Phase 2 was determined by earlier experiments with the previous game releases. During Phase 3, the pupils were solving posttests. This activity also took one school hour (45 minutes) for each group.

Activities in Phases 1 and 3 started with reading the instructions for solving tasks out loud to the whole group and reviewing three presolved examples. The rest of the time was devoted to solving tasks. A simple scoring scheme was used for the evaluation of the pre- and posttests. Each successfully solved task was worth one point. Zero points were given for empty or incorrectly solved tasks.

According to the assessment of knowledge described earlier, we set the metric to measure minimal standard knowledge as follows:

1. The pupil must solve one question from each category of tasks successfully.
2. In addition, the pupil must solve successfully at least one question with the DIP objective.

Phase 2 also started with short instructions for playing the game. Next, pupils were given time to transform their avatar as they wished and to familiarize with commands in the game environment. Then, participants created their first program in the sandbox area. The first debriefing was carried out after that. Next, pupils started solving the actual tasks. When most of pupils in the group entered a new difficulty level, the playing was paused and a new debriefing was initiated.

The data were gathered and processed with Microsoft Excel (2016) and the "R" software environment (2016). jMetrik (Meyer, 2016) was used for IRT analysis.

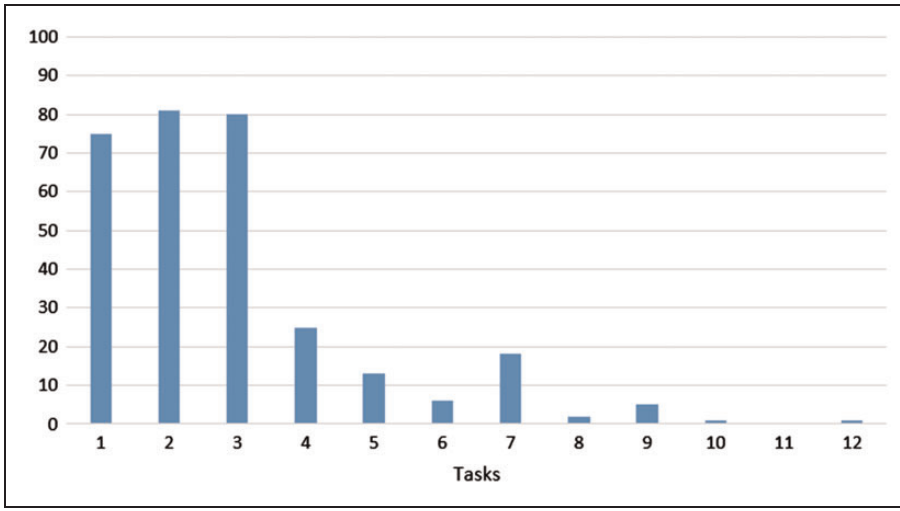

Figure 7. Frequencies of the correct answers in the pretest.

Results and Discussion

The scores undoubtedly show progress of knowledge and the benefits of the proposed application of the educational game. Paired-samples t test was conducted to compare participants' scores between the pretest and posttest. There was a significant difference in the scores for pretest ($M = 3.07$, $SD = 1.43$) and posttest ($M = 8.79$, $SD = 2.22$); ($t(99) = 30.77$, $p < .001$). In this and all other statistical tests, we used a confidence factor of 0.95. Figure 7 shows the frequencies of the correct answers in the pretest. Figure 8 shows the frequencies of the correct answers in the posttest.

Tasks 1, 4, 7, and 10 are related to the COP objective. Tasks 2, 5, 8, and 11 are related to the CRP objective. Tasks 3, 6, 9, and 12 are related to the DEP objective. In addition, Tasks 4, 5, 10, 11, and 12 require the pupils to achieve the DIP objective.

The first three tasks in the pretest were simple enough to allow more than 70% of pupils to solve them intuitively. They also did not include any of the complex programing concepts and are deliberately meant to be easy. These results show that pupils did not have problems with understanding the construct of the task. A few pupils also solved correctly some of even more difficult tasks in the pretest, which was not expected. We assume that the reason for that is the pseudocode used in the construct of the task, which is simple enough even for novices. Nevertheless, results also indicate that none of the participants had any prior knowledge of programing skills.

Almost all pupils answered the simplest questions correctly in the posttest. They were less successful with tasks 5, 6, 9, and 11. Tasks 5 and 11 had DIP and

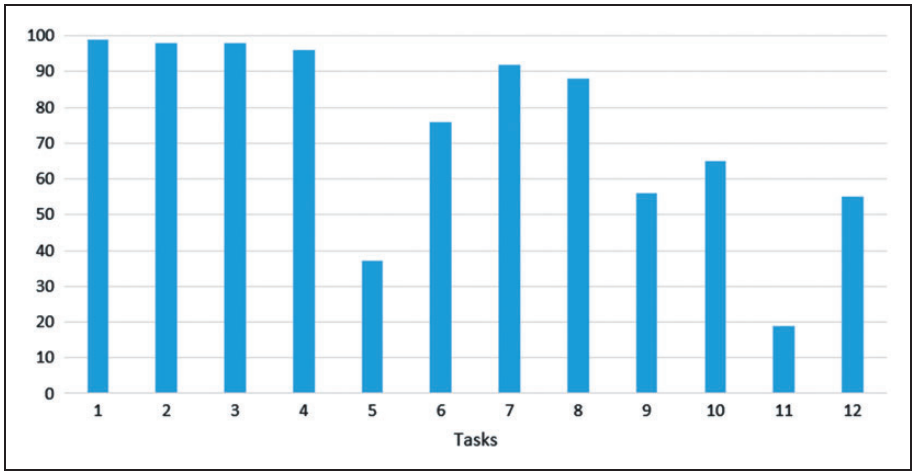

Figure 8. Frequencies of the correct answers in the posttest.

Table 1. Item Response Theory Classifications of Our Tasks.

	Difficulty (0%–100%)			
	Hard (0%–50%)	Moderate (50%–85%)	Easy (85%–100%)	Total
Discrimination				
Poor (<0.1)			2	1 item
Fair (0.1–0.3)	11		1, 3, 4	4 items
Good (>0.3)	5, 12	6, 9, 10	7,8	7 items
Total	3 items	3 items	6 items	12 items

CRP objectives combined, which evidently makes it more demanding. Tasks 6 and 9 did not include a DIP objective but probably caused confusion with some pupils. In Task 6, they were puzzled by the possibility of the object traveling across marked cells more than once. In Task 9, the starting point was positioned higher on the grid than the finish of the path. Some pupils were confused by that, which made their objects travel in the wrong direction.

Difficulty and discrimination of the tasks was evaluated with IRT analysis. Difficulty measures the percentage of pupils that solved a given task correctly. Discrimination measures how well a given correctly solved task predicts the overall pupil's performance. The results are shown in Table 1. We used Miranda Parkers et al. (2016) classifications. Almost all tasks have predominant fair and good discrimination. However, some tasks have a level of difficulty

lower than expected. Nevertheless, the assessment shows evidence of validity and is suitable for the varying ability levels of the participants in the experiment.

According to our metric, 25% of the pupils have met the minimum standards on the pretest and after the experiment, 96% of the pupils have met them on posttest. The reasons why 4% of pupils failed to achieve a set goal are difficult to analyze. Their results in the posttest do not correlate with the scores reached in the Aladdin game or with their assessment marks in science subjects. The causes for that might be found in the live course environment where all sources of disruptions were impossible to predict. In some cases, the local teacher would distract pupils, making them feel less willing to solve tests. Two groups, with that issue, achieved a below average mean ($M = 7.45$, $M = 7.5$). To see if this had any impact on the results, a one-way ANOVA was contacted on the participant's posttest of each group. The analysis of variance showed that there was no significant difference in scores between groups, $F(8, 91) = 1.23$, $p = .28$. The most possible cause for the failure was that we were not successful in motivating all participants to engage fully in solving tasks in the posttest.

Further, we analyzed the progress of knowledge based on learning objectives from the curricula, which is shown in Figure 9. The average number of successfully solved tasks for each category was used for analysis. The scores obtained from tasks with DIP objective were normalized because the number of tasks was greater than the number of tasks in the other categories. The biggest progress was observed in tasks with COP and DEP objectives. Less progress is visible in tasks with CRP and DIP objectives. Tasks where those two objectives combined had caused the greatest cognitive demand. In these tasks, the program has to be written from the scratch, a repeating pattern has to be found in the problem, and loops and subprograms have to be used.

Pearson's correlation was measured between the scores of the posttest with several preconditioned factors. Based on the results, pretest scores are in weak

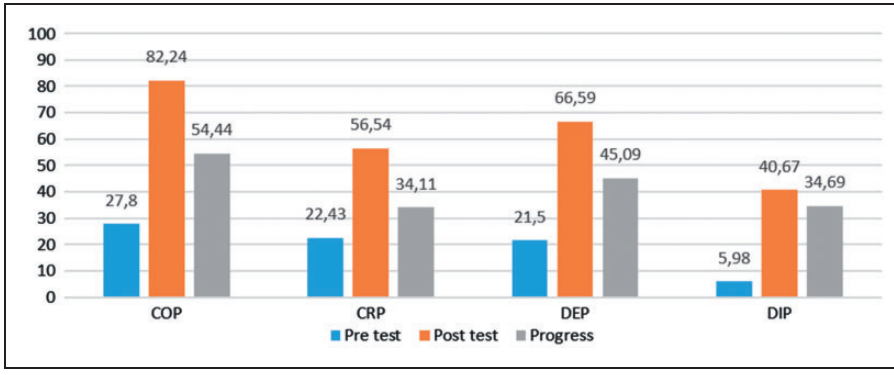

Figure 9. Progress of knowledge according to learning objectives.

correlation with posttest scores ($r = .55$, $p < .01$). Posttest scores are also in weak correlation with the pupils' grade in Mathematics ($r = .52$, $p < .01$) and Natural Sciences ($r = .43$, $p < .01$). The posttest scores are uncorrelated with the amount of time the pupils spent playing with the entertainment games ($r = .01124$, $p = .09116$). However, the posttest scores are correlated strongly with the scores achieved in the Aladdin game ($r = .82$, $p < .01$). That outcome shows that the solving the problems in the game has significant impact on the knowledge acquired. To elaborate this, we also observed the correlation between the results of the game with the scores of posttest for each learning objective. The strongest correlation was found with the DIP objective ($r = .75$, $p < .01$). This link between educational game and DIP objective is very promising. It could suggest that with appropriate application of an educational game, certain more demanding learning objectives can be met at the beginning of the Course.

We also observed from the gender perspective. An independent-samples t test was conducted to compare average score of boys and girls. There was insignificant difference in the average scores on the posttest for boys ($M = 9.05$, $SD = 2.15$) and the girls ($M = 8.63$, $SD = 2.34$); ($t(99) = .47$, $p = .64$). According to the results, we can assume that the Aladdin game is equally suitable for both ganders.

The results of the experiment showed that achieving even more demanding standards of observed learning objectives with the proposed application of educational games is possible. In this way, teachers are able to plan pupils' learning experience better. We believe that knowing the reason and purpose for using educational games and when to stop using them is very important because it can maximize games' learning benefits. Educational games should be used at intervals and only for a short period of time. Each period should consist primarily of problems with fundamental programing concepts and cognitive, more demanding ones at the end. An extended application of educational games would also build some kind of ''learning momentum,'' which would be strong enough to maintain a pupil's interest in programing even though they tackle more difficult problems. The ''learning momentum'' could be utilized to bring learning experience from the game to other programing environments. The cycle could be used several times during the course in order to ''feed the learning momentum.'' It is also possible to continue learning activities in a game simulation environment. Blockly library could be used instead of control panels (Google Project, 2015). Later, Scratch4opensim (Rossenbaum, 2016) and C# could be used to manipulate with objects in the virtual world.

Conclusion

This study was set out to evaluate the effectiveness of the educational game ''Aladdin and his flying carpet'' in the Introductory Computer Science

Course, which was designed based on other research findings and built with all contextual and pedagogical consideration in mind. This study provides us with an example of a successfully implemented educational game and its assessment of knowledge based on observed learning objectives from curricula in the sixth grade of primary schools.

The major findings of this study are summarized as follows. Pupils (96%) have successfully reached a minimum standard of knowledge just by playing the educational game. The biggest progress is noticeable with the ''complete a program'' objective and the smallest with the ''create a program'' objective. The least successfully solved tasks were the ones where the ''create a program'' and ''divide a problem'' objectives were joined. The differences between pupils at the beginning of the study, which are derived from their assessment marks in science subjects and their time spent with playing entertainment games, were balanced out. The progress was visible with all the participants and was correlated strongly with the results achieved in playing the game. The strongest correlation was found between the scores of the game and the ''divide a problem'' learning objective, which also invites more research. Another notable finding was that the boys were only slightly more successful than the girls and the difference in results is statistically insignificant.

Based on the findings, the study also proposes an extended application for use in class, which will be the topic of future work. Besides for learning fundamental programing concepts, the educational games can be also used for more demanding objectives. We hypothesize that playing the proper educational games would also generate some kind of ''learning momentum,'' which would be strong enough to allow teachers to switch learning environments, where additionally more challenging problems could be addressed. The cycle, where games and other learning environments are to be switched, can be used several times in the Course allowing momentum to gain strength. If the teaching methods of the whole cycle were to be designed according to the constructivist learning theory where pupils learn by interacting with their environment and peers, learning momentum could be measured with number, content or significands of communications between participants.

There are some other considerations of the research that we will pursue in the future. The usability test could be augmented with appropriate methods that would allow us to locate possible weak spots. We would compare scores from the pre- and posttests with other already validated assessments. We would introduce the interviews and questionnaires to measure the level of motivation. In the experiment, we perused only the minimal learning objectives. More demanding standards of knowledge can be observed with the evaluation. Furthermore, the transfer of knowledge from the icon-based programing used in the game to the pseudocode syntax used in pre- and posttests could also be measured.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

References

Allcancode, Inc. (2016). *Allcancode - run marco!* Retrieved from https://www.allcancode.com/

ARNES. (2015). *Costum virtual servers*. Retrieved from http://www.arnes.si/storitve/splet-posta-strezniki/streznik-po-meri.html

Bobrow, D. G., Feurzeig, W., Papert, S., & Solomon, C. (2016). *Logo (programming language)*. Retrieved from https://en.wikipedia.org/wiki/Logo_(programming_language)

Carroll, J. (1998). *Minimalism beyond the nurnberg funnel*. Cambridge, England: MIT Press.

CeeBot. (2016). *CeeBot: Have fun programming*. Retrieved from http://www.ceebot.com/ceebot/index-e.php

Code.org. (2016a). *Code.org - Flappy*. Retrieved from http://studio.code.org/flappy/1

Code.org. (2016b). *Code.org - Frozen*. Retrieved from http://studio.code.org/s/frozen/stage/1/puzzle/1

CodeCombat. (2016). *CodeCombat* - Learn how to code by playing a game. Retrieved from http://codecombat.com/

CodeMonkey Studios Ltd. (2016). *CodeMonkey*. Retrieved from http://www.playcodemonkey.com

COLOBOT. (2016). COLOBOT: A new 3D real time game of strategy and advanture. Retrieved from http://www.ceebot.com/colobot/index-e.php

Computer Science Teachers Association. (2015). *Computer science teachers association*. Retrieved from http://csta.acm.org/

Cunniff, N., & Taylor, R. (1987). *An empirical study of novices' program comprehension – Graphical vs. textual representation*. Norwood, NJ: Ablex Publishing Corp.

Dijkstra, E. (2016). *On the cruelty of really teaching computing science (EWD-1036)*. Retrieved from http://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1036.PDF

Discover Sensors. (2016). *Discover sensors*. Retrieved from http://www.discoversensors.ie/junior_certificate/%20reflection_stems1

Egenfeldt-Nielsen, S. (2007). *The educational potential of computer games*. Copenhagen, Denmark: Continuum Press.

Egenfeldt-Nielsen, S. (2015). *eLearn magazine: What makes a good learning game? Going beyond edutainment*. Retrieved from http://elearnmag.acm.org/archive.cfm?aid=1943210

Elliott Tew, A. (2010). Assessing fundamental introductory computing concept knowledge in a language independent manner. *A dissertation presented to the academic faculty*. School of Interactive Computing Georgia Institute of Technology.

Felicia, P. (2009). *Digital games in schools: A handbook for teachers*. Brussels, Belgium: European Schoolnet.

Fletcher, G., & Lu, J. (2009). Education: Human computing skills: Rethinking the K-12 experience. *Communications of the ACM - Inspiring Women in Computing*, *52*, 23–25.

Freitas, S. d. (2007). *Learning in Immersive worlds – A review of game-based learning*. LOnon, England: JISC.

Gee, J. (2005a). Good video games and good learning. *Phi Kappa Phi Forum*, *85*(2), 33.

Gee, J. (2005b). What would a state of the art instructional video game look like? *Innovate: Journal of Online Education*, *1*(6), Article 1.

Gee, J. (2008). ''Learning and Games.'' The ecology of games: Connecting youth, games, and learning. In K. Salen, D. John & T. Catherine (Eds.), *MacArthur foundation series on digital media and learning* (pp. 21–40). Cambridge, MA: The MIT Press.

Goldman, K., Gross, P., Heeren, C., Herman, G., Kaczmarczyk, L., Loui, M., & Zilles, C. (2010). Setting the scope of concept inventories for introductory computing subjects. *ACM Transactions on Computing Education*, 10, 1–29.

Google Project. (2015). *Blockly games*. Retrieved from https://blockly-games.appspot.com/

Google Project. (2016). *Blockly - Google developer*. Retrieved from https://developers.google.com/blockly/

Honey, M., & Hilton, M. (2011). *Learning science through computer games and simulations*. Washington, DC: The National Academies Press.

Jakoš, F., & Lokar, M. (2015). A language independent assessment of programming concepts knowledge. Ljubljana, Slovenia: ISSEP.

Kiili, K. (2005). *On educational game design*. Tampere, Finland: Tampere University of Technology.

Kolb, D., & Fry, R. (1975). Toward an applied theory of experiential learning. In C. Cooper (Ed.), *Theories of group process*. London, England: John Wiley.

Lewis, L., & Williams, C. (1994). Experiential learning: Past and present. *New Directions for Adult and Continuing Education*, *62*, 5–16.

c, I. (2016). *Second life*. Retrieved from http://secondlife.com/

Lode, H., Franchi, G., & Frederiksen, N. (2012). *Learning games for programming*. (Master thesis). Copenhagen, Denmark: IT University Copenhagen.

Maragos, K., & Grigoriadou, M. (2007). Designing an educational online multiplayer. *Proceedings of the Informatics Education Europe II Conference IEEII*, 322–331.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2010). Learning computer science concepts with scratch. *Proceedings of the Sixth international workshop on Computing education research*, 69–76.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2011). Habits of programming in scratch. *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, 168–172.

Members of the R Development Core Team. (2016). *The R project for statistical computing*. Retrieved from http://www.r-project.org

Meyer, J. P. (2016). *jMetrik*. Retrieved from http://www.jmetrik.com

Microsoft. (2016). *Microsoft excel*. Retrieved from https://products.office.com/excel

Microsoft Research. (2016). *CodeHunt*. Retrieved from https://www.codehunt.com/

Ministry of Education, Science and Sport. (2015). *The curriculum for the optional elective subject*. Retrieved from http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/os/devetletka/program_razsirjeni/Racunalnistvo_izbirni_neobvezni.pdf

MIT Media Lab. (2016). *Scratch*. Retrieved from https://scratch.mit.edu

Moodle. (2016). Moodle - Open source learning platform. Retrieved from https://moodle.org

MULTI-DIMENSIONAL. (2016). *CodeSpells*. Retrieved from http://codespells.org/

Niato, C. (2016). *Lightbot*. Retrieved from http://lightbot.com/

Norving, P. (2016). *Teach yourself programming in ten years*. Retrieved from http://norvig.com/21-days.html

Opensimulator. (2016). Opensimulator. Retrieved from http://opensimulator.org

Ostrovsky, I. (2016). *Robozzle*. Retrieved from http://www.robozzle.com/

Papert, S. (1998). Does easy do it? Children, games, and learning. *Game Developer magazine*, p. 88.

Parker, M., Guzdial, M., & Engleman, S. (2016). Replication, validation, and use of a language independent CS1 knowledge assessment. *Proceedings of the 2016 ACM conference on international computing education research* (pp. 93–101). New York: ACM.

Peake, A. (2016). *CODEHERO* – A game that teaches how to make games. Retrieved from https://codeherogame.wordpress.com/

Programming Education Tools Group, U. o. (2015). *greenfoot.org*. Retrieved from greenfoot.org

Rico, M., Martínez-Muñoz, G., Alaman, X., Camacho, D., & Pulido, E. (2011). Improving the programming experience of high school students by means of virtual worlds. *International Journal of Engineering Education*, *27*(1), 52–60.

Robins, A., Rountree, J., & Rountree, N. (2002). Identifying the danger zones: Predictors of success and failure in a CS1 course. *Inroads (the SIGCSE Bulletin)*, *34*, 121–124.

Robocode Home. (2016). Robocode Home. Retrieved from http://robocode.sourceforge.net/

Robot Battle – Wikipedia. (2016). Robot Battle – Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Robot_Battle

Rossenbaum, E. (2016). *Scratch4opensim*. Retrieved from https://scratch4opensim.wikispaces.com/

Sedighian, K., & Sedighian, A. (1996). Can educational computer games help educators learn about the psychology of learning mathematics in children? *In 18th Annual Meeting of the International Group for the Psychology of Mathematics Education – The North American Chapter, Florida, USA (1996)*.

Shaffer, D., Squire, K., Halverson, R., & Gee, J. (2005). Video games and the future of learning. *Phi Delta Kappan*, *87*, 104–111.

Sharp, L. (2012). Stealth learning: Unexpected learning opportunities through games. *Journal of Instructional Research*, *87*, 42–48.

Sloodle. (2016). Sloodle. Retrieved from https://www.sloodle.org/

Soloway, E., & Spohrer, J. (1989). *Studying the novice programmer*. New Jersey, NJ: Lawrence Erlbaum.

THE A-GAMES PROJECT, University of Michigan. (2016). *The A-games project*. Retrieved from http://gamesandlearning.umich.edu/a-games/

The Alice Project. (2015). *The alice project*. Retrieved from alice.org

UNESCO. (2013). *Final recommendations towards knowledge societies*. Geneva: Author.
Vygotsky, L. (1978). *The development of higher psychological processes*. Cambridge, England: Harvard University Press.
Ward, B., & Bell, T. (2009). *Using virtual world programming languages to teach computer science*. Retrieved from http://www.cosc.canterbury.ac.nz/research/reports/HonsReps/2009/hons_0907.pdf
Wastiau, P., Kearney, C., & Berghe, W. (2009). *How are digital games used in schools?* Brussels, Belgium: European Schoolnet.
Weintrop, D., & Wilensky, U. (2015). Using commutative assessments to compare conceptual. *Proceedings of the eleventh annual International Conference on International Computing Education Research*, 101–110.
Wikipedia. (2016). *Aladdin*. Retrieved from https://en.wikipedia.org/wiki/Aladdin
Wilson, A. K., Bedwell, W. L., Lazzara, H. E., Salas, E., Burke, C. S., Estock, L. J., . . . Conkey, C. (2009). Relationships between game attributes and learning outcomes. *Simulation and Gaming*, *40*, 217–266.
Winslow, L. E. (1996). Programming pedagogy – A psychological overview. *SIGCSE Bulletin*, *28*, 17–22.
Zagami, J. (2008). *Seeing is understanding: The effect of visualisation in understanding programming concepts*. Brisbane, Australia: Queensland University of Tehnology.

Author Biographies

Franc Jakoš is a Computer Science teacher at Janka Glazerja Primary School. Recently, he also started teaching at Selnica ob Dravi Primary School. His work interests are introducing games into educational process and designing and validating language independent assessments. His hobby interest is breeding sport horses for jumping.

Dr. Domen Verber is professor at Faculty of Electrical Engineering and Computer Science. He is actively involved with projects of Real-Time Systems Laboratory at Institute of Informatics. His main research topics are Software development methodologies, Real-time systems, Embedded computing systems, Programming languages and compilers, High Performance computing, Computer games, Machine learning, and artificial intelligence.