Computers in Human Behavior 52 (2015) 200–210

Contents lists available at ScienceDirect

Computers in Human Behavior

journal homepage: www.elsevier.com/locate/comphumbeh

A new way of teaching programming skills to K-12 students: Code.org

Filiz Kalelioğlu

Baskent University, Faculty of Education, Ankara, Turkey

ARTICLE INFO

Article history:

Keywords:
Improving classroom teaching
Programming and programming languages
Elementary education

ABSTRACT

This study attempts to investigate the effect of teaching code.org site on reflective thinking skills towards problem solving. More specifically, this study attempts to investigate whether there is a gender difference in terms of students' reflective thinking skills towards problem solving. This triangulation study was conducted with 32 primary school students. The quantitative part of the study was conducted in pre-test/post-test comparison design of quasi-experimental design. The scores of reflective problem solving skills were gathered through the reflective thinking skill scale towards problem solving and the students' performances in the code-org site were examined. In the qualitative part of the research, after the five-week experimental process, focus group interviews were conducted with ten students and a reflection paper from the IT teacher was analysed. According to the *t*-test results, teaching programming to primary school students in the code.org site did not cause any differences in reflective thinking skills towards problem solving. However, there is a slight increment in the means of female students' reflective thinking skills towards problem solving over the males' reflective thinking skills towards problem solving. On the other hand, qualitative data provided more information about the students' experiences. Students developed a positive attitude towards programming, and female students showed that they were as successful as their male counterparts, and that programming could be part of their future plans.

1. Introduction

In the 21st century, people are expected to not only be consumers, but also producing individuals. With this as a basis, there is a need to train children at a younger age, as they are digital natives, born into a technological world, seem to readily adapt to new technologies, and use it easily. Whilst it would seem a little easier to train this generation as producers, what do we need to do? There is a need to start changing education programmes according to the developments in computer science (Grout & Houlden, 2014; Jones, 2013), as the very nature of computer sciences is dynamic, flexible and innovative. Then, teachers, administrators, parents, and society in general have to give sufficient importance to computer science in order to create innovators. They need to be educated in a way that stimulates high order thinking skills. Moreover, they have to be provided with problem solving opportunities, to be encouraged to think creatively, to make decisions and to reflect upon their solutions. To realise these goals, there is a need to help children at a young age to develop a love for computer sciences.

Worldwide, many countries have updated their computer science curriculum to teach students from a younger age (e.g. Bargury et al., 2012; Bers, Flannery, Kazakoff, & Sullivan, 2014; Grgurina, Barendsen, Zwaneveld, van Veen, & Stoker, 2014; Grout & Houlden, 2014; Jones, 2013; Kalelioğlu, Gülbahar, Akçay, & Doğan, 2014; Lee, Martin, & Apone, 2014; Repenning, Webb, & Ioannidou, 2010). There are many emerging applications that can be used to teach students computer science, robotics programming, and drag and drop blocks. In addition to these, programming events are held globally which aim to teach children how the computer works, to endear them to computer science and to increase high-order thinking skills. In fact, the process of teaching computer programming to students is a difficult subject area. In older-age students, they lack logical reasoning and algorithmic thinking, and may have problems in programming courses. When algorithmic rules of logic and syntax of programming languages are coupled within a course, unresolved issues may arise (Kristi, 2003; Robins, Rountree, & Rountree, 2003). Therefore, to help children learn how a computer works in a fun way, it is extremely important that these skills are developed in children of a younger age.

When studies about this issue are analysed, there are some research studies found that address the effect of teaching programming to children on their higher level thinking skills, such as algorithmic thinking, critical thinking, creative thinking, and problem solving. Although empirical support for the cognitive benefits of these tasks has been slow to emerge (Denner, Werner, & Ortiz, 2012), self-perceptions of the students reveal that they have

E-mail address: filizk@baskent.edu.tr

http://dx.doi.org/10.1016/j.chb.2015.05.047
0747-5632/

already gained at this point. Almost all of the children in the research studies developed a positive attitude to learning computing (Bers et al., 2014; Fessakis, Gouli, & Mavroudi, 2013; Kalelioğlu & Gülbahar, 2014; Keren & Fridin, 2014; Rogozhkina & Kushnirenko, 2011). Upon deeper examination of the literature, there have been no studies that have addressed students' reflective thinking skills towards problem-solving. Moreover, none address the experiences of students with code.org, a drag and drop visual programming platform launched in 2013. Currently, the effects of teaching programming with code.org site still remain unclear in the literature. It is therefore necessary to explore how code.org site influences students' learning experiences and their reflective thinking skills towards problem solving. This would provide a better understanding regarding how to apply blocky coding to promote students' reflective thinking skill towards problem solving, and as such, such a study would contribute to the literature.

In light of this, this study attempts, at a macro level, to investigate the effect of teaching code.org site on reflective thinking skills towards problem solving. More specifically, this study attempts to investigate whether there is a gender difference in terms of students' reflective thinking skills towards problem solving. At a micro level, this study addresses learning experiences in the code.org site and more specifically, this study aims to investigate students' thoughts about programming and the code.org site. For this purpose, the following questions were examined:

1) What is the effect of teaching programming on reflective thinking skill towards problem solving of primary school students?
2) Is there a gender difference on reflective thinking skill towards problem solving of primary school students?
3) What is the performance of the students on code-org site?
4) What do they think about programming and the code-org site?
5) What were the reflections of the IT teacher on teaching programming to the students?

1.1. Reflective thinking skill towards problem solving

Reflective thinking is a part of the critical thinking process referring specifically to the processes of analysing and making judgments about what has happened. Reflective thinking skill towards problem solving is essential during complex problem-solving situations, since it provides students with an opportunity to step back and think about how they actually solve problems and how a specific set of problem solving strategies is appropriated for completing their goal (Lin, Hmelo, Kinzer, & Secules, 1999). Moreover, according to Kızılkaya and Aşkar (2009), reflective thinking could be best observed in the problem-solving process and mainly composed of reflective habits such as questioning (searching answer for the problems), reasoning (examining cause-effect relationships), and evaluation (stepping back and thinking about how problems are actually solved).

Writing program code is similar with general problem solving processes, as often it utilises the same steps. There are some prerequisites to writing programs, including applying in-depth reading skills and meta-cognitive skills in order to judge how to solve the programming problem effectively and clearly. Moreover, programmers need to manage their problem-solving processes, apply a programming methodology, correct any programming errors, check their program output and think deeply and reflect about their programming solutions. Therefore, this on-going process will lead students to manage their programming processes, question their decisions and actions and explore alternative solutions in order to develop the quality of their programs (Havenga et al., 2013).

According the results of Bergin, Reilly and Traynor's study (2005), students who perform well in programming use more meta cognitive–management strategies than lower-performing students. Akcaoglu and Koehler (2014) designed a study to provide experimental support for the effects of learning game-design on students' problem-solving skills. When students in the control group were compared with the treatment group, the children who attended the game-design learning programme showed a significant increase in their problem-solving. According to the study of Liao and Bright (1991), most of the students were positive and favoured the computer programming group over the control groups; in addition, students having computer programming experiences scored higher on various cognitive-ability tests compared to students without prior programming experience. In the study of Fessakis et al. (2013) concerning the dimensions of problem solving using computer programming by five and six year old kindergarten children, the children were found to enjoy the engaging learning activities and had opportunities to develop mathematical concepts, problem solving and social skills. Research by Zhang, Liu, Ordóñez de Pablos, and She (2014) found that teaching programming by using Alice, an innovative 3D programming environment, makes it easier to help students learn fundamental programming concepts in the context of creating animated movies and video games. The results of the study by Denner et al. (2012), provided that game construction, including both design and programming activities, can support the learning of computer science concepts. Regarding the design activities, Zhang, Ordóñez de Pablos, and Zhu (2012) found that 3D virtual world may influence team leaning at the individual level including coding language skills, and personal creativity and at team level providing rich social resources, social networks, and efficient team dynamics.

1.2. Teaching programming to children

If you can't create programming logic and use algorithmic thinking skills, developing programs can be very difficult. A lack of operational sequence, the use of invalid programming language syntax and poor use of problem solving strategies can turn attempts at programming into a chaotic endeavour. Programming processes can be taught using different instructional tools and techniques. Zhang et al. (2014) asserted that most students are more comfortable with learning programming through visual presentation (diagrams, video, animation), verbal explanation and by discovering things on their own. Moreover, with drag and drop type applications, younger students can not only learn computer science and informatics concepts, but also develop higher order thinking skills. As Fessakis et al. (2013) stated, computer programming can be seen as an important competence for the development of higher-order thinking, such as algorithmic problem solving skills. Research of Liu, Cheng, and Huang (2011), revealed that students who perceived a flow experience, applied trial-and-error, learning by example, and analytical reasoning strategies to learn computational problem solving skills. Similarly, Keren and Fridin (2014), found that children enjoyed robotics programming and results showed that their performances on geometric thinking and metacognitive tasks were improved when programming.

Teaching programming to children can be traced to the 1960s with Logo programming language that was written in 1968. This programming language was "designed to provide a conceptual foundation for teaching mathematical and logical ways of thinking in terms of programming ideas and activities" (Feurzeig & Papert, 2011, p. 487). By using Logo programming, Papert (1971) has advanced that logo can be used for developing mathematical concepts and it has strong advantages over classical topics for children. As Lye and Koh (2014) stated that after Logo programming

language, the use of programming tools for teaching programming to children were not broadly investigated until the availability of easy to use visual programming languages.

1.3. Gender issues in programming

Experiences show that boys are developing programming skills easier than girls in higher education (Kiss, 2010). Plass et al. (2007) noted that girls and women, in particular, show less interest in computer science and programming. Nevertheless this situation is quite different at the primary education level. Regarding gender issues in teaching programming, Bruckman, Jensen, and DeBonte (2002) conducted a study for comparing the female and male students in a programming environment. According to the results, there is no gender difference affecting programming performance of the children in terms of their level of programming achievement. Similarly, in a study of Owston, Wideman, Ronda, and Brown (2009), they concluded that there were no gender differences in the game development task. Moreover, Vos, van der Meijden, and Denessen (2011) conducted a study with 235 elementary school students about constructing drag and drop memory games. According to the results, they found no significant gender differences for perceived competence, interest or use of deep learning strategies.

As a suggestion for attracting the interest of female students towards programming, Bruckman et al. (2002) offered that educators wishing to increase gender equity in technical skills should focus on strategies to foster interest among girls. More specifically, Hutchinson, Moskal, Cooper, and Dann (2008) suggested that educational software like Alice or other drag and drop applications, can be used in introductory programming for teaching programming and aiming to eliminate gender differences in programming attitudes. Similarly, Baytak and Land (2011) offered that implications from their study are drawn for the potential role of drag and drop programming applications as a vehicle to engage more girls in computer science. Supporting this fact, Kelleher and Pausch (2006) found an increase in girls' interest in learning programming when they were required to create visual stories with specific software.

1.4. Code.org

With today's advancing technology and applications, many programming platforms have evolved in terms of blocky coding such as Scratch, Alice, Blockly, and Kodu. These online activities use blocky, a visual programming language, where you drag and drop blocks together in order to generate code. Code.org is a blocky coding platform used for this purpose. Launched in 2013, code.org presents the opportunity for students to learn computer science through drag and drop programming. The site, available in 34 languages, is structured as a game, and consists of self-directed tutorials, posters and video lectures by famous people such as Bill Gates and Mark Zuckerberg as a way to help inspire students to look at computer sciences. With the help of these educational materials, students can learn the logic of algorithm, 'if' conditions, variables, loops and functions. The site has courses that teach computer science to students. In this study, the researcher focused on a free course known as 'K-8 Intro to Computer Science', that aims not only to teach computer science in a fun, collaborative, and creative way, but also to teach computational thinking, problem solving and programming. The course is designed to motivate students and educators to continue learning computer science as a way to improve real-world relationships, connections, and life.

The course is suitable for students from kindergarten through to 8th grade, and beyond. The content of the 20-h course includes online and offline tutorials, as well as offline teacher-facilitated lessons. The key concepts taught on the course are: what is computer science?; what is a computer scientist?; how to be a responsible computer scientist; the applications of computer science; basic understanding of binary; how to debug; how the internet works; programming concepts; sequencing, loops, conditionals, functions, functions with parameters, variables, computational thinking, decomposition, patterns, abstraction, and algorithms. This course uses a blended learning approach to teaching computer science, which means that students learn from a mix of online, self-guided activities and unplugged activities with teacher-led activities that use no computer at all. Students receive prizes for completing all of the stages (Code.org, 2014).

Students can also see their own learning process, monitor which levels they have completed, which are still to be completed, and what trophies they earned in which concept. Moreover, they can see a summary of performance, such as "You've earned 7 out 27 'Concept Mastery' trophies. Where you left off: Stage 2 Puzzle 5". Additionally, students can see their performance visually. Fig. 1, shows a sample screen of a learning process in terms of the computer science course and the concept mastery trophies gained in these activities.

2. Method

2.1. Research design

This study was carried out in triangulation design. In the triangulation design, researchers use the quantitative and qualitative approaches during the same timeframe with equal weight. "This design is used when a researcher wants to directly compare and contrast quantitative statistical results with qualitative findings or to validate or expand quantitative results with qualitative data" (Creswell & Clark, 2007, p.62).

The quantitative part of the study was conducted in pre-test/post-test comparison design of quasi-experimental design that lacks the element of random assignment. The independent variable of the study was teaching programming to primary school students via code.org site. The dependent variables of the study were reflective problem solving skills. The scores of reflective problem solving skills were gathered through the Reflective Thinking Skill Scale Towards Problem Solving (RTSSTPS), and the students' performances in the code-org site were examined. In the qualitative part of the research, after the five-week experimental process, focus group interviews were conducted with 10 programming students and a reflection paper from the IT teacher was analysed. The design of the study is presented in Table 1.

2.2. Participants

The participants were 32 primary school students from the 4th year attending a computer course at a private school in Turkey. Of the 32 students, there were 17 females and 15 males. All students were aged ten years old. The course was taught for one hour per week.

2.3. Context and process

Teaching programming via code.org site continued for a period of five weeks with 4th year primary school students. The reason for selecting code.org site for this study, is that it has many valuable aspects when compared with other programming tools. Most importantly, the site has pedagogical materials and well-organized activities from simple to complex in terms of teaching programming. In addition, the teacher is able to monitor the activities of his/her entire class, what levels they completed, and

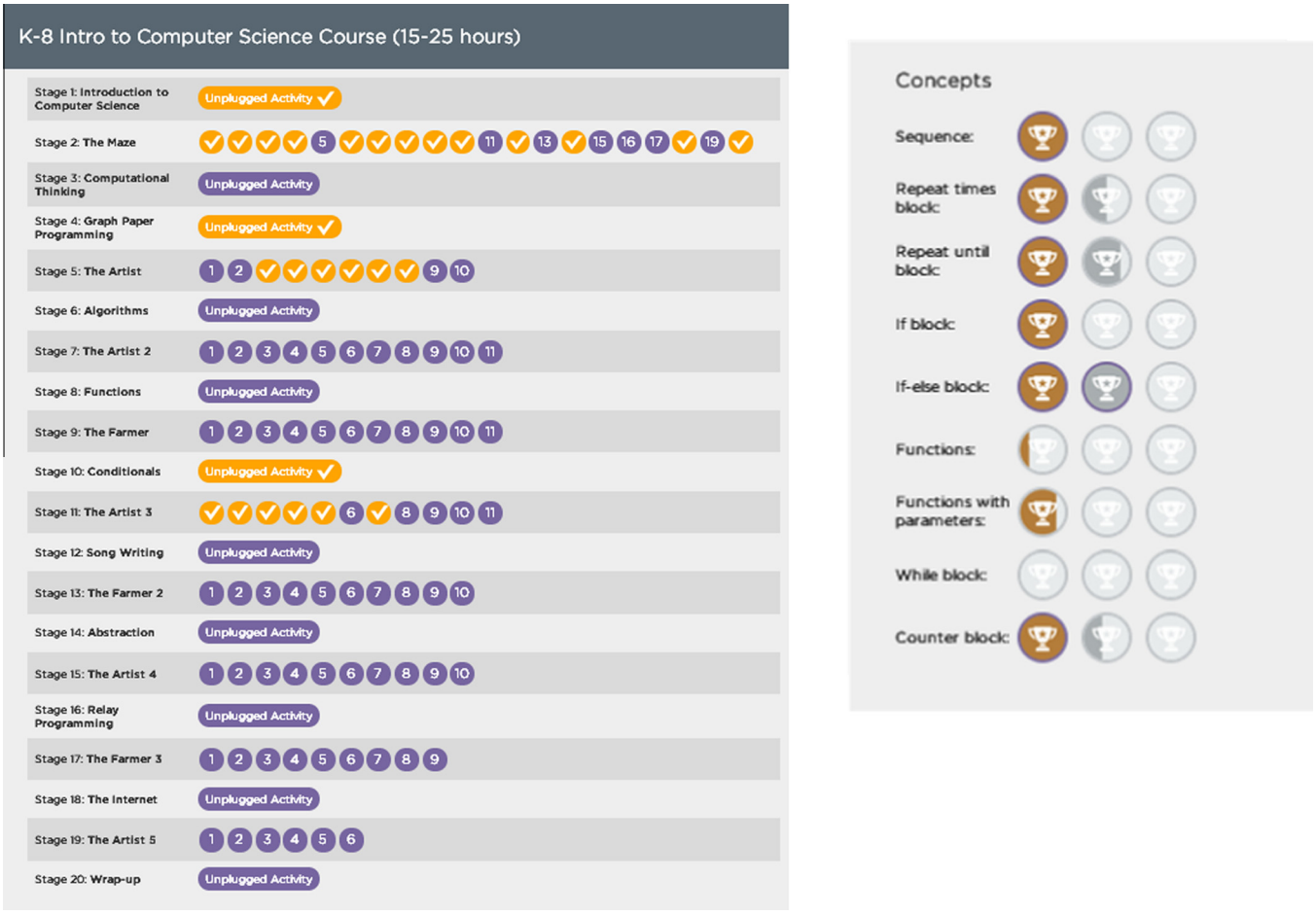

Fig. 1. A screenshot of self-learning progress from code.org site.

Table 1
Design of the study.

N	Pre-test	Treatment (5 weeks)	Post-test
32	RTSSTPS	T	RTSSTPS Focus group interview

time spent on the activities. This platform makes it easier for providing guidance and feedback to the students. The code.org site works as a learning-programming management system.

Before the start of the study, Institutional Review Board (IRB) approval and consent was received from parents for their children to participate. In the first introductory lesson, the importance of programming, code.org site's contribution to mathematical thinking and problem solving skills were explained to the students. A five-week course syllabus was created by the IT teacher and implemented respectively. For the first week, topics such as 'what is programming', 'what is software', 'where can software be used', 'what does it do', and 'why programming is important' were discussed and the "Code Stars" video on YouTube was monitored. Then, the scale for reflective thinking skills towards problem solving was administered to the students. After completion of the scale, students entered the learn.code.org site and become members using class codes assigned to them.

In the second week, the video in the first stage of Introduction to Computer Science was monitored and then the first two levels of the Maze stage was completed with students and the logic of the site was explained. Then students were allowed to study on other levels of Maze individually at their own pace.

In the third week, the students were taught about the site and the monitoring of self-progress. To achieve this, the students were taught to view their progress dashboard in order to see their completed levels. According to this, dark green boxes mean that the level is 'completed, perfect'; light green boxes mean that the level is 'completed, but using too many blocks'; purple boxes mean that the level is 'tried, but not completed', and lastly, a white box means that the level is 'not tried yet'. On the site, students receive trophies depending on their success at each stage. It was explained to the students that the dark yellow icon is a 'gold trophy', the light yellow was a 'bronze trophy', and the silver icon was a 'silver trophy'.

Fig. 2. The third week's applications.

After the above, the Maze puzzle was continued. In Fig. 2, the 'repeat' block for the 6th puzzle is shown, together with the 'repeat until' block (10th puzzle) and the 'if' block (14th puzzle).

For the fourth week, the stage three: Computational Thinking, stage four: Graph Paper Programming and stage five: The Artist were introduced to the students. The videos in stages three and four were showed to the students, and stage five was covered in more detail. The drawing square in the third puzzle, changing colour in the fifth puzzle, drawing square and triangle in the sixth puzzle, drawing glasses in the seventh puzzle and going backward commands were then taught (Fig. 3).

For the fifth week, stages six: Algorithms and seven: The Artist 2 were introduced to the students. After watching a video on stage six, stage seven was explained in more detail to help the students. In detail, drawing and transformation triangle in the second puzzle, repeating in the seventh puzzle, drawing square in the ninth puzzle and drawing and transformation square in tenth puzzle were completed with the help of the IT teacher (Fig. 4).

After five weeks of tuition and all applications completed, RTSSTPS was re-administered to the students as a post-test. Moreover, students went on studying on the code.org site in order to complete the unfinished puzzles, both at home and in the classroom.

2.4. Data collection tools

In this study, multiple data collection tools were used in order to bring answers the research questions. For research questions one and two, an instrument was used; for research question three, students' performance was analysed from the code.org site, for research question four and its related sub-questions, a focus group interview process was conducted; and for the last question, a reflection paper from the IT teacher was analysed.

2.4.1. Instrument

As a quantitative measure, the Reflective Thinking Skill Scale Towards Problem Solving of Kızılkaya and Aşkar (2009) was used in this study. This measurement tool can be used for finding out primary education school students' reflective thinking skill towards problem solving. The scale was applied to 339 (174 female, 165 male) primary education students and statistical analysis was then performed. After factor analysis, the Cronbach Alpha coefficient was found to be 0.83. In this inventory, there are 14 items with three factors; questioning, reasoning and evaluation.

2.4.2. Focus group interview

After completion of the five-week research process and implementation of the scale, the IT teacher chose five students from each class, according to their different levels of interest and success in code.org programming. The researcher then conducted a focus group interview with a structured interview form. The following questions were asked to the students:

a. Did you have difficulty with the problem solving process?
b. Could you solve problems in different ways? Or could you try to solve?
c. Was the use of code.org site easy? Did you face any problems within the platform?
d. What was your favourite stage on code.org site?
e. What was your least favourite stage on code.org site?
f. What are the benefits of programming on code.org site?
g. Do you like programming?
h. What do you want to learn more about in programming? Do you want to improve yourself?

Of the 10 students who participated to the focus group interview, there were four females and six males. Interviews were recorded with the aid of a voice recorder. The total duration of the interviews was 10:51 and 11:17 min respectively.

2.4.3. Performance on code.org site

Students' performance on the code.org site was analysed from the teacher's account, and the students' learning process put into a table to show the status of them.

2.4.4. Reflections of the IT teacher

The reflections of the IT Teacher were a free format document that consists of three paragraphs explaining the process of teaching programming in code.org site and thoughts and suggestions.

2.5. Data analysis

To reveal whether there was a difference between pre-test and post-test, paired samples T-test was calculated with the data gathered from the scale. In addition to this, to test whether there was a gender difference between pre-test and post-test, descriptive statistics was calculated. The data gathered from each focus group interview were firstly transcribed verbatim, and then content analysis process was applied. Each interview was read more than once, interviews were coded and a frequency table created, the emerging themes were identified, harmonisation of codes and themes were examined. The artefacts, progress of each students and other information from code.org site was put into the table to compare the students' performance. Finally, content analysis process was applied to the reflection paper of the IT teacher.

3. Results

3.1. Effect of Code.Org programming on reflective thinking skill towards problem solving of students

Before performing statistical analyses, normality of pre-test and post-test of RTSSTPS was tested. According to the Shapiro–Wilk test, the data for the pre-test ($p = .319$) and post-test of RTSSTPS ($p = .090$) were distributed normally. After testing the assumptions, t-test for pre-test and post-test of RTSSTPS was calculated (Table 2). According to these results, there were no differences between pre- and post-test results of the students' reflective thinking skill towards problem solving [$t(31) = -0.51$, $p > .05$]. Moreover, t-test for pre-and post-test between sub-scales was also conducted. There were only slight increases in the means of the factors of 'questioning' and 'evaluation'. In other words, programming in code.org site did not cause any differences in the reflective thinking skill towards problem solving of the primary school students. This result may show that programming in code.org site may not have an effect on the reflective thinking skill.

3.2. Gender differences on reflective thinking skill towards problem solving of primary school students

Means and standard deviations of the female and male students gathered from pre- and post-test of RTSSTPS are shown in Table 3. The means of the pre-test of female students was 4.16 while the means of the post-test was 4.23. Moreover, the means of the pre-test of the male students remained the same with the post-test at 4.03. It can be said that there is a slight increment in the female students' reflective thinking skills towards problem solving over the males' reflective thinking skills towards problem solving.

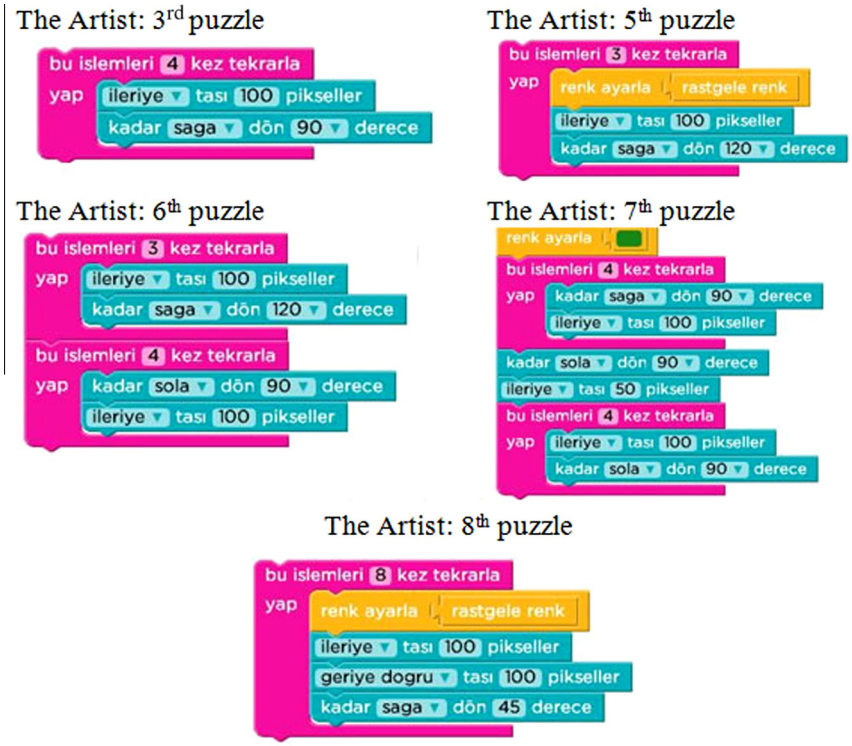

Fig. 3. The fourth week's applications.

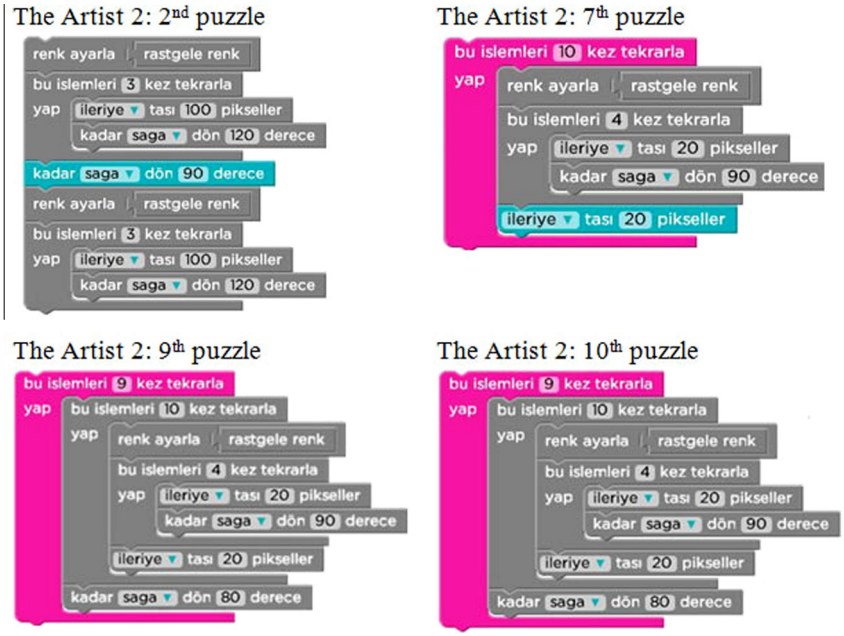

Fig. 4. The fifth week's applications.

3.3. Students' performance in the Code.Org site

To analyse the students' performance in the code.org site, the stages and levels that students completed, and the trophies collected, were counted and tabulated. For the five weeks of class activities, the students' performance on the code.org site was much higher than for their normal classroom activities; meaning that students tried to complete more stages and levels at their own pace. As Table 4 depicts, all students completed the Maze (19.53/20), Artist (9.25/10) and for Artist 2 (8.84/11) most levels of the stage were completed. In addition to this, they completed more than half of the levels of the Farmer stage (6.66/11), a few levels of Artist 3 (3.06/11), and attempted Farmer 2 (0.25/10), Farmer 3 (0.03/9) and Artist 5 (0.22/6) stages. No students completed any levels of Artist 4. The reasons for this may be that each stage became more challenging in terms of programming, therefore students needed more time and more support to complete these stages.

Table 2
T-test results of pre-test and post-test of RTSSTPS.

Test	Mean	N	Standard deviation	*df*	*t*	*p*
Pre-test	4.10	32	0.57	31	−0.51	0.617
Post-test	4.13	32	0.61			
Pre-test – questioning	4.05	32	0.60	31	−1.30	0.202
Post-test – questioning	4.16	32	0.65			
Pre-test – evaluation	4.05	32	0.77	31	−0.39	0.697
Post-test – evaluation	4.09	32	0.70			
Pre-test – reasoning	4.38	32	0.53	31	1.72	0.096
Post-test – reasoning	4.22	32	0.62			

Table 3
Means and standard deviations of pre-test and post-test of RTSSTPS.

Gender	Pre-test			Post-test		
	N	Mean	Standard deviation	N	Mean	Standard deviation
Female	17	4.16	0.58	17	4.23	0.63
Male	15	4.03	0.58	15	4.03	0.59
Total	32	4.10	0.57	32	4.13	0.61

When the performance of female and male students were examined, there is just a slightly difference in the performance based on their average completion of stages and levels (Fig. 5).

When the rewards (trophies) were examined, how many trophies taken at what subjects were counted, as shown in Table 5. Female students won 16 gold trophies and the males won 17; female students won 37 silver trophies and the males won 48; female students won 42 bronze trophies and the males won 23. In summary, female students won 95 (51.91%) of trophies, while male students won 88 (48.09%).

Students' average completions of the stages may seem similar; however there is a difference in favour of the females when it comes to trophies. The site awards trophies to students according to quick and accurate level completion. These findings can be interpreted that females completed the programming activities more quickly and accurately than the male students. However, these findings cannot be generalised; this situation may need to be examined in further detail.

3.4. Thoughts of the students about programming and code.org site

3.4.1. Thoughts about problem solving process

When the thoughts of the students about problem solving were examined, the answers of the students were divided into two themes: 'difficulty' and 'assistance' (Table 6). Under the theme of 'difficulty', which had a total of 11 answers, nine students said that they sometimes had difficulties, one student said that some stages were hard, and one student stated not having much difficulty. Under the theme of 'assistance', which had a total of 10 answers, five students said that they got help from other students, while the others were those that provided help those working at stages previous to theirs. In this respect; one student declared that "*Sometimes, we explained what we had done to those who couldn't make progress. We helped those left behind when we moved forward.*" Differently, one student stated that "*In fact, there are hints. It writes at the bottom [of the page] as 'do like that for levelling up'. Sometimes I have benefitted from that.*"

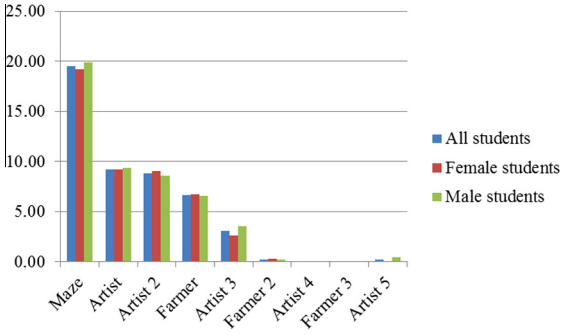

Fig. 5. The graphs of females' and males' stage completion performance.

3.4.2. Thoughts about solving problems in different ways

When the thoughts about solving problems in different ways were analysed, the answers of the students were divided into two themes: 'new ways' and 'not new ways' (Table 6). Under the theme of 'new ways', with a total of nine answers, all nine students said that they used new ways to solve their problems, three students expressed that they used different movement commands and two students used different triangle drawings according to the stages. Under the theme of 'not new ways', one student said that he did not try new ways because he wanted to finish the stage and pass to the next level as soon as possible. Regarding this question, one student stated that *"I thought how I could pass through the stages in a different way where I couldn't succeed."* Differently one declared that *"I sometimes tried to solve problems in a different way than my friends. But I was trying to be fast to pass through more stages."*

3.4.3. Ease of Use of code.org site

When students' thoughts about the usage of the code.org site were examined, all of the students' answers were positive (Table 6). Nine students said that code.org site was very easy (to use), two students explained that the site was like a game and was very enjoyable, and lastly, one student said that the site was very good due to having popular characters (Angry Birds, Zombies etc.) that they had used previously. About the ease of use of the code.org site, one student expressed that "*The code.org site was entertaining, sounded like a game.*" Similarly, another student explained "*in particular, we were enjoying it more because it has characters in the games we have played before in our tablets.*"

3.4.4. Favourite stages

According to the answers given for the students' favourite stages on code.org, two themes were revealed as 'favourite stages'

Table 4
Average completion of stages and levels that students had completed.

Students	Stages								
	Maze 20 levels	Artist 10 levels	Artist 2 11 levels	Farmer 11 levels	Artist 3 11 levels	Farmer 2 10 levels	Artist 4 10 levels	Farmer 3 9 levels	Artist 5 6 levels
Female	19.18	9.18	9.06	6.71	2.65	0.29	0.00	0.00	0.00
Male	19.93	9.33	8.60	6.60	3.53	0.20	0.00	0.07	0.47
Total	19.53	9.25	8.84	6.66	3.06	0.25	0.00	0.03	0.22

and 'reasons' (Table 6). Under the theme of 'favourite stages', artist 3 (5), the maze (2), artist (2), farmer (2), artist 2 (1), farmer 2 (1) and artist 4 (1) were the mostly favoured stages. For the question as to why they preferred those stages to others, four students expressed that they most enjoyed these stages, whilst four other students explained that those stages were the most challenging and informative, and lastly, two students stated that those stages contained the better characters. Regarding this question, one student expressed that "*I like [the stage that has the] zombie [character]. There are things more challenging and more informative."* Similarly another student declared *"I like the artist stage because it was more difficult and made me use my brain"*. Lastly, one student stated that *"I have learned something called 'pixel' that I have never known. I liked it."*

3.4.5. Least-favourite stages

According to the answers given for the least-favourite stages of code.org, with similarity to the previous answers, two themes were revealed; 'least-favourite stages' and 'reasons' (Table 7). Under the theme of 'least-favourite stages', artist 3 (4), the maze (2), artist (1), farmer (1) were the least favoured. For the question as to why they did not favour those stages to the other ones, five students expressed that those stages were easy, three students explained that those stages were more challenging, one student said that those stages were boring and lastly, one male student stated that those stages had female characters.

Regarding this question, one student stated that "*I did not like [the stage that has] Angry Birds. It came a little easier*." Differently, one student expressed that "*I did not like the farmer stage. It felt boring. The characters were female*". Lastly, one student explained that "*Some stages of the Zombie was difficult. I was annoyed that I could not complete the stage, so I did not like it*."

3.4.6. Benefits of using code.org site

When the thoughts about benefits of using code.org site were questioned (Table 7), 10 students stated that they learned programming, seven expressed that they improved their mathematical and geometrical knowledge, two students had learned to find solutions through less steps, two students stated that they had a chance to observe results, one had improved computer literacy, whilst one mentioned improved cognition, one stated that programming helped create a love of mathematics, one student expressed learning to think logically, and lastly one expressed learning to find directions.

Regarding benefits of the programming in code.org site, one student stated that "It helps us to get good grades in our mathematics exam." Similarly, other one expressed "*Normally, some may not like math; when this program is used, they enjoyed math*". Another explained that "*. . .I was completing math test slowly before using the program. After using the program, I started working faster. One declared that "It taught the using of less commands to reach a solution.*" Lastly, one explained that "*Actually, I'm not very good at math. I made some mistakes in the process. I made logic mistakes. But I've learned to use my logic when we used this program.*"

3.4.7. Thoughts about programming

When students were asked whether they liked programming or not, all of the students responded positively.

3.4.8. Their thoughts on their Future programming

On the question of the students' thoughts about future programming (Table 7), 10 students said that they wanted to learn more about programming, four wanted to teach others how to program, three wanted to use other programming platforms, two wanted to use code.org site at home, one wanted to finish all of the stages on code.org, and lastly, one student wanted code.org

Table 5
Concept mastery Trophies those students had won.

Trophy	Sequence		Repeat times block		Repeat until block		If block		If else block		Functions									
											Function with Parameters		While block		Counter block				Fem. total	Male total	Total
F	M	F	M	F	M	F	M	F	M	F	M	F	M	F	M	F	M	F			
M	F	M	F	M	F	M	F	M	F	M	F	M	F	M	F	M	F	M			
Gold	0	0	0	0	0	1	0	0	14	13	0	0	2	3	0	0	0	0	16	17	34
Silver	4	5	4	8	11	14	10	10	2	2	6	8	0	1	0	0	0	0	37	48	89
Bronze	13	9	12	7	5	0	6	4	0	0	3	1	2	1	1	1	0	0	42	23	65
Total	17	14	16	15	16	15	16	14	16	15	9	9	4	5	1	1	1	0	95	88	183

Table 6
Thoughts about programming and the code-org site.

Thoughts about problem solving process	
Themes	# of incidents
Thoughts about problem solving process	
Difficulty	
Sometimes had difficulty	9
Did not have much difficulty	1
Some stages were hard	1
Assistance	
Get help from other students	5
Helped to other students	5
Thoughts about Solving Problems in Different Ways	
New ways	9
Different movement commands usage (turn left, jump forward etc.)	3
Different triangle drawings	2
Not new ways	1
Ease of use of code.org site	
Easy usage	
Very easy	9
Enjoyable/like a game	2
Good for having popular characters	1
Favourite stages in the code.org site	
Favourite Stages	
Artist 3	5
The maze	2
Artist	2
Farmer	2
Artist 2	1
Farmer 2	1
Artist 4	1
Reasons	
Enjoyable	4
More challenging/informative	4
Having good characters	2

Table 7
Thoughts about programming and the code-org site – 2.

Thoughts about problem solving process	
Themes	# of incidents
Least-favourite stages in the code.org site	
Least-favourite Stages	
Artist 3	4
The maze	2
Farmer	1
Artist	1
Reasons	
Easy	5
More challenging	3
Boring	1
Had female characters	1
Benefits of using code.org site	
Teaching programming	10
Improving mathematical and geometrical knowledge	7
Finding solutions with less steps	2
Having chance to observe the results	2
Improving computer literacy	1
Improving cognition	1
Helping to love math	1
Thinking logically	1
Finding directions	1
Students' thoughts on their Future Programming	
Learn more about programming	10
Want to teach others how to program	4
Want to use other programming platforms	3
Want to use code.org at home	2
Want to finish all stages on code.org	1
Want code.org to become popular in others schools	1

become popular in other schools. Regarding the last question, one student stated that "*I want to complete all stages of code.org site.*" Others declared "*I want all schools to use code.org.*", "*I want to teach others programming.*" and "*I want to learn [programming] sites like code.org".*

3.5. Reflections of the IT teacher

According to the reflections of the IT teacher, the answers were composed of two main themes: code.org site/its features and the process of programming. For the first theme, she found the code.org site to be very useful and helpful, and added that having language support and the fact that the site is popular worldwide are very important. Moreover, she found it easy to sign into the site for both students and teacher, and also to create classes to make groups online. If students forgot their passwords, a password reset is performed via the teacher's account, so even if students consistently forget their passwords, it does not create problems for the teacher. Also, teachers can monitor the progress of the students; which levels they complete, which trophies they receive, and which level they are currently studying. Teachers can prepare certifications for successful students to motivate them. She found the videos and notes between the stages to be very useful. Moreover, teachers can prepare the content of the lesson by using these notes. Another important feature of the site that she mentioned was that students can individually progress in the site by using these informative course notes at their own pace. She found these notes and videos to be valuable materials due to the fact that they were provided by successful names from the IT sector.

According to theme for the process of programming, she mentioned that students favoured the first stages because of the Zombie and Angry Bird characters that they were already familiar with. Moreover she added that this was a good way to start getting used to code.org. According to the observations of the IT teacher, her students had difficulties with the drawings in the Artist stages because of the fact that mathematical calculations were used a bit more. Students were quite pleased with their individual progress; they asked each other at what stage they are at and they have said that they will continue to do puzzles at home. Through games, they often calculated in which direction and angle they would return. Thus, both their geometrical and directional information is quite evolved. Moreover, they learned algorithm and basic programming unwittingly by solving the puzzles. They learned loops, if structure and variables with the help of repeat blocks, condition blocks and counter blocks. Additionally, she expressed that as students completed each stage and obtained trophies, they were more willing to continue. Some children, who were previously not interested in the lesson, participated and showed amazing progress and were obviously very fond of this site (code.org).

4. Discussion and conclusion

The main purpose of this research study was to explore the effects of code.org programming on 4th grade primary school students' reflective thinking skills towards problem solving skills. According to the t-test results, programming in code.org site did not cause any differences in the reflective thinking skill towards problem solving of primary school students. There were only slight increases in the means of the factors of 'questioning' and 'evaluation'. The self-perception about students' reflective thinking skills was found to be above average, but the reason for the slight increase in the results can be down to the duration of programming activities. More importantly, the results may be caused by

the number of students participating in the study. Similarly, according to the quantitative results of Kalelioğlu and Gülbahar (2014), programming in the Scratch platform did not cause any significant differences in the problem solving skills of the primary school students. There was a slight improvement in the students' self-confidence in their problem-solving ability.

When the performances of female and male students were examined both quantitatively and qualitatively, there was a slight difference in the means of pre-test and post-test of RTSSTPS and in the performance between females' and males' average completion of stages and levels they had completed. This result showed that female and male students performed equally and both completed as many stages as possible. However, the female students obtained more trophies than their male counterparts. This result is consistent with other research results (e.g. Bruckman et al., 2002; Owston et al., 2009; Robertson, 2012; Vos et al., 2011). Correspondingly, Robertson (2012) found similar results, stating that unlike the literature on gender issues of classroom games projects, the results of the study indicated that females' games score more highly than males', particularly on skills relating to storytelling.

Moreover, students benefit from programming courses not only quantitatively but also qualitatively. To support this fact, all participating students thought that using code.org site teaches programming and improves mathematical and geometrical knowledge. Regarding this result, Fessakis et al. (2013) also found that children liked the programming activities and had opportunities to develop mathematical concepts, problem solving and social skills. According to the other result, all of the students stated that they liked programming in code.org site and wanted to learn more about programming. This result is also in parallel with Navarrete's (2013) study, concerning teaching programming to children. According to this study, creative thinking process in a game creation learning approach may provide learners with a rich and enjoyable learning experience, with authentic technology usage and the provision of insightful learning. Moreover, according to the results of Bers et al. (2014), kindergartners were interested in programming and able to learn many aspects of robotics, programming, and computational thinking.

When thoughts of the students were investigated, most of the students had difficulties when programming, and they would seek help from each other. All of the students tried to solve problems in different ways, they found the site easy, and half of the students liked the Artist 3 stage, whilst the others did not. When examined more deeply, students failed to give appropriate directional commands to characters in the code.org site. This issue is also underlined within the study of Fessakis et al. (2013). According to their students, "the students also faced great difficulty in determining which command could move the ladybug in the appropriate direction, looking for a button with an arrow pointing 'down'" (p. 93). When programming with objects, this problem can be critical for students and some precautions should be taken at this point. Visual-spatial abilities of the students should be enhanced with various activities, such drag and drop programming activities, robotics, simulation games, drawings, 3-D modelling etc. Moreover, teachers could bring a robot to the class; students could then give it commands and observe the results.

The reflections of the IT teacher were also found to be very valuable. Code.org site is very suitable for students to learn programming; because of the many features such as being easy to use, having motivating factors for students (popular characters, the chance to obtain trophies, hints to find the solutions etc.), being enjoyable, having an opportunity to monitor self-progress, and the provision of embedded teaching materials. Moreover, it was seen very clearly that students learned computer science concepts on this platform while playing in an enjoyable way. Those who want to teach programming by using the code.org site can benefit from the teaching progress in this study.

As a conclusion about the reflections of the students and the teacher, Code.org site has motivating features, helpful instructional materials and lesson plans inside, can be a good option for those who want to teach programming to novices and to support students to gain the basis of computer science. Moreover, experiences gained from this study underline the support of the children by utilisation of such programming platforms. Therefore, the early integration of programming in education is essential, due to the fact that programming activities contribute to student development and gives them a chance to write their own games in order that they can become productive people, and not just consumers. Whatever the result, students should be introduced as soon as possible to programming activities, planned in a way to consider the development, needs and expectations of the students.

This study is primarily limited by its sample size and classes attending the computer course. A larger sample would have benefitted the results of the study. Moreover, the study is limited by the activities implemented during just five weeks. The activities should be enhanced and the time spent for teaching programming to children should also be extended for deeper understanding of the fundamentals of programming. In this regard, the conceptual achievements of the students can be also evaluated. To put it more explicitly, the contribution of such platforms used for teaching programming into learning programming concepts underlining the cognitive issues can be investigated as further research.

References

Akcaoglu, M., & Koehler, M. J. (2014). Cognitive outcomes from the Game-Design and Learning (GDL) after-school program. *Computers & Education, 75*, 72–81.

Bargury, I. Z., Muller, O., Haberman, B., Zohar, D., Cohen, A., Levy, D., et al. (2012). Implementing a new computer science curriculum for middle school in Israel. *Proceedings of Frontiers in Education Conference (FIE)*, 1–6.

Baytak, A., & Land, S. M. (2011). Advancing elementary-school girls' programming through game design. *International Journal of Gender, Science and Technology, 3*(1), 243–253.

Bergin, S., Reilly, R., & Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. *Proceedings of the First International Workshop on Computing Education Research*, 81–86.

Bers, M. I., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education, 72*, 145–157.

Bruckman, A., Jensen, C., & DeBonte, A. (2002). Gender and programming achievement in a CSCL environment. In *Proceedings of the conference on computer support for collaborative learning: Foundations for a CSCL community* (pp. 119–127).

Code.org (2014). *Teach our K-8 intro to computer science*. <http://code.org/educate/20hr>.

Creswell, J. W., & Clark, V. L. P. (2007). *Designing and conducting mixed methods research*. Thousand Oaks, CA: Sage.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education, 58*, 240–249.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87–97.

Feurzeig, W., & Papert, S. A. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments, 19*(5), 487–501.

Grgurina, N., Barendsen, E., Zwaneveld, B., van Veen, K., & Stoker, I. (2014). Computational thinking skills in dutch secondary education: Exploring teacher's perspective. In *Proceedings of the 9th workshop in primary and secondary computing education* (pp. 124–125).

Grout, V., & Houlden, N. (2014). Taking computer science and programming into schools: The Glyndŵr/BCS Turing Project. *Procedia – Social and Behavioral Sciences, 141*(25), 680–685.

Havenga, M., Breed, B., Mentz, E., Govender, D., Govender, I., Dignum, F., et al. (2013). Metacognitive and problem-solving skills to promote self-directed learning in computer programming: Teachers' experiences. *SA-eDUC Journal, 10*(2), 1–14.

Hutchinson, A., Moskal, B., Cooper, S., & Dann, W. (2008). The impact of the Alice curriculum on community college students' attitudes and learning with respect to computer science. In *Proceedings of the annual meeting of the American society for engineering education.* Pittsburgh, PA.

Jones, S. P. (2013). *Computing at school in the UK*. <http://research.microsoft.com/en-us/um/people/simonpj/papers/cas/computingatschoolcacm.pdf>.

Kalelioğlu, F., Gülbahar, Y., Akçay, S., & Doğan, D. (2014). Curriculum integration ideas for improving the computational thinking skills of learners through programming via scratch. In *Local proceedings of the 7th international conference on informatics in schools: Situation, evolution and perspectives* (pp. 101–112).

Kalelioğlu, F., & Gülbahar, Y. (2014). The effect of teaching programming via scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education, 13*(1), 33–50.

Kelleher, C., & Pausch, R. (2006). Lessons learned from designing a programming system to support middle school girls creating animated stories. *Proceedings of Visual Languages and Human-Centric Computing*, 165–172.

Keren, G., & Fridin, M. (2014). Kindergarten Social Assistive Robot (KindSAR) for children's geometric thinking and metacognitive development in preschool education: A pilot study. *Computers in Human Behavior, 35*, 400–412.

Kiss, G. (2010). A comparison of programming skills by genders of hungarian grammar school students. *Proceedings of the Symposia, and Workshops on Ubiquitous Autonomic and Trusted Computing*, 24–30.

Kızılkaya, G., & Aşkar, P. (2009). The development of a reflective thinking skill scale towards problem solving. *Eğitim ve Bilim, 34*(154), 82–92.

Kristi, A. -M. (2003). *Problems in learning and teaching programming – a literature study for developing visualizations in the Codewitz-Minerva Project*. <http://www.cs.tut.fi/~edge/literature_study.pdf>.

Lee, I., Martin, F., & Apone, K. (2014). Integrating computational thinking across the K–8 curriculum. *ACM Inroad, 5*(4), 64–71.

Liao, Y. C., & Bright, G. W. (1991). Effects of computer programming on cognitive outcomes: A meta-analysis. *Journal of Educational Computing Research, 7*(3), 251–266.

Lin, X., Hmelo, C., Kinzer, C. K., & Secules, T. J. (1999). Designing technology to support reflection. *Educational Technology Research & Development, 47*(3), 43–62.

Liu, C. C., Cheng, Y. B., & Huang, C. W. (2011). The effect of simulation games on the learning of computational problem solving. *Computers & Education, 57*, 1907–1918.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61.

Navarrete, C. C. (2013). Creative thinking in digital game design and development: A case Study. *Computers & Education, 69*, 320–331.

Owston, R., Wideman, H., Ronda, N. S., & Brown, C. (2009). Computer game development as a literacy activity. *Computers & Education, 53*(3), 977–989.

Papert, S. (1971). *Teaching children to be mathematicians vs. teaching about mathematics*. <http://publications.ai.mit.edu/ai-publications/pdf/AIM-249.pdf>.

Plass, J. L., Goldman, R., Flanagan, M., Diamond, P., Dong, C., Looui, S., et al. (2007). *RAPUNSEL: How a computer game design based on educational theory can improve girls' self-efficacy and self-esteem*. <http://steinhardtapps.es.its.nyu.edu/create/courses/2176/reading/AERA_07_Rapunsel_Plass_etal.pdf>.

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proceedings of the 41st ACM technical symposium on computer science education* (pp. 265–269).

Robertson, J. (2012). Making games in the classroom: Benefits and gender concerns. *Computers & Education, 59*, 385–398.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137–172.

Rogozhkina, I., & Kushnirenko, A. (2011). PiktoMir: Teaching programming concepts to preschoolers with a thinking and metacognitive development in preschool education: A pilot study. *Computers in Human Behavior, 35*, 400–412.

Vos, N., van der Meijden, H., & Denessen, E. (2011). Effects of constructing versus playing an educational game on student motivation and deep learning strategy use. *Computers & Education, 56*(1), 127–137.

Zhang, J. X., Liu, L., Ordóñez de Pablos, P., & She, J. (2014). The auxiliary role of information technology in teaching: Enhancing programming course using alice. *International Journal of Engineering Education, 30*(3), 560–565.

Zhang, J. X., Ordóñez de Pablos, P., & Zhu, H. (2012). The impact of second life on team learning outcomes from the perspective of IT capabilities. *International Journal of Engineering Education, 28*(6), 1388–1392.