Source: Informatics in Education - An International Journal

Informatics in Education - An International Journal

Location: Lithuania

Author(s): Yasemin GÜLBAHAR, Filiz KALELIOĞLU

Title: The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective
The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective

Issue: 1/2014

Citation style: Yasemin GÜLBAHAR, Filiz KALELIOĞLU. "The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective". Informatics in Education - An International Journal 1:33-50.
https://www.ceeol.com/search/article-detail?id=123695

Informatics in Education, 2014, Vol. 13, No. 1, 33–50

The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective

Filiz KALELIOĞLU[1], Yasemin GÜLBAHAR[2]

[1] *Baskent University, Faculty of Education Baglica Campus Ankara, Turkey*

[2] *Ankara University, Department of Informatics Kecioren Campus Ankara, Turkey*

e-mail: filizk@baskent.edu.tr, gulbahar@ankara.edu.tr

Received: January 2014

Abstract. Computer programming is perceived as an important competence for the development of problem solving skills in addition to logical reasoning. Hence, its integration throughout all educational levels, as well as the early ages, is considered valuable and research studies are carried out to explore the phenomenon in more detail. In light of these facts, this study is an exploratory effort to investigate the effect of Scratch programming on 5th grade primary school students' problem solving skills. Moreover, the researchers wondered what 5th grade primary school students think about programming. This study was carried out in an explanatory sequential mixed methods design with the participation of 49 primary school students. According to the quantitative results, programming in Scratch platform did not cause any significant differences in the problem solving skills of the primary school students. There is only a non-significant increase in the mean of the factor of "self- confidence in their problem solving ability". When the thoughts of the primary students were considered, it can be clearly stated that all the students liked programming and wanted to improve their programming. Finally, most of the students found the Scratch platform easy to use.

Keywords: programming and programming languages; elementary education; improving classroom teaching.

1. Introduction

Courses with names like "Introduction to Programming" are very traditional and are often compulsory courses for many students studying in higher education. With the establishment of competencies for 21st century Information and Communication Technology (ICT), in recent years the programming concept has moved into secondary and even primary schools. In terms of competencies for the 21st century, we require the skills needed to survive in the information age in a global society, and these are computer skills, information technology, media, and information literacy skills (Nelson, 2009). Ornelas-Marques and Marques (2012) also pointed out the importance of skills for the 21st centu-

ry and they stated that youth are lacking these skills. Similarly, Papadopoulos and Tegos (2012) stated that "… research in the field of computer science education has highlighted that many students lack problem solving and computational thinking skills" (p. 180).

It is obvious that informatics and ICT courses are reshaped having "programming skills" as a competence for students in the early ages. Many educators face difficulties in the process of teaching students programming skills and logic. So these difficulties bring about the questioning of our teaching methodology. Another question that comes to mind is if it is difficult at the higher education level, what should be the most appropriate strategy to deal with primary and secondary grade students? Emphasising the importance of high-level competences like critical thinking, problem solving and autonomy, Ornelas-Marques and Marques (2012) pointed to Scratch programming language to overcome this problem They stated that when learners use a programming language like Scratch "... they develop all these skills, and they soon find problems that they immediately have (feel the need) to solve in order to advance with the project" (p. 61).

With these questions in mind, this study tried to explore the effectiveness of using Scratch in the process of teaching children programming in terms of contribution to problem solving skills.

Scratch is software that can be used to program interactive stories, games, and animations and share all the creations with others in the online community (http://scratch.mit.edu/). Hence, Scratch helps students to think creatively, reason systematically, and work collaboratively; all of which are essential skills required for the 21st century. Scratch was developed at the MIT Media Lab and is free of charge. As defined by Lamb and Johnson (2011), "In computer software, scratching refers to reusable pieces of code that can easily be combined, shared, and adapted. Students can create stories, games, art, music, animations, and much more" (p. 64). With the use of Scratch, users can create projects using downloaded or web based software, and then upload their projects to the Scratch web site for sharing.

1.1. *Lessons Learned from Previous Research Studies*

Firstly, some research studies carried out with teens were considered. Owing to the fact that pre-service teachers find it difficult to master the syntax of programming languages in general, a computer practice course was redesigned using Scratch in order to promote computational and creative thinking and to overcome possible difficulties by Kim, Choi, Han and So (2012). The researchers used Scratch for pre-service teachers to create an area for their innovative ideas and to platform pre-service teachers' thinking processes. Kim, Choi, Han and So (2012) concluded that "Unlike the traditional computer programming languages, Scratch helped pre-service teachers focus on what they could do with programming languages. Acquiring programming concepts has become implicit rather than explicit" (p. 971). Furthermore, the researchers stated that pre-service teachers should utilise Scratch both from a pedagogical and a design point of view.

Another research conducted for higher education by Xinogalos (2012) aimed at investigating whether the knowledge acquired in the context of an introduction to fundamental OOP concepts with the microworlds is transferable afterwards to Java. The findings

showed that "… the most fundamental OOP concepts with the microworld objectKarel had a positive impact on students' achievements, both in comprehending the corresponding concepts and achieving better results when moving to Java1 (p. 272). In their study, Cegielski and Hall (2006) investigated the predictors of object-oriented programming performance and based on extensive data analysis, they found that each of the constructs of theoretical value belief, cognitive ability, and personality exhibits a predictive relationship with OO programming performance. Hence, cognitive ability seems to be one of the most important predictors in programming.

Theodorou and Kordaki (2010) designed and implemented a 4-level computer game using the tools provided by Scratch and in parallel they also exploited the JIGSAW collaborative method. The researchers aim was to provide a learning environment for high school students on the concept of variable programming. The researchers concluded that since the programming procedure is dependent on constructing blocks of simple commands, instead of writing in an interactive environment, the Scratch environment could be helpful to students.

Lewis (2011) used Scratch to design learning environments and evaluated individual work with pair work by comparing two highly collaborative learning environments to isolate aspects of pair programming that are significant for supporting students' understanding, attitudes and interest in computer science, and pace. The researcher concluded that "There are obvious benefits of pair programming over environments without support and collaboration" (p. 129). It can be concluded that Scratch has some benefits for teens and this environment can be a powerful source for teens. The potential of the Scratch environment is also underlined by Lee (2011), where the researcher concluded that educators can benefit from Scratch by developing creative, entertaining, interdisciplinary curriculum materials, and in this way learners can unleash their imagination in a more meaningful and engaging way.

As a second step, research studies completed with the children as participants were provided. Lai and Yang (2011) conducted a study to investigate the effect of visualised programming on the learners' problem-solving abilities and logical reasoning skills. Their sample was 6th graders who took a Scratch programming course over one semester. The researchers found out that the effect on problem-solving abilities was significant, especially with reason of prediction, whereas there was no significant effect on logical reasoning skills. They concluded that "… integrating visualised programming learning and problem solving strategies can enhance the learner's problem-solving skills" (p. 6940). Underlining the importance of problem solving ability, the researchers also suggested that the teachers of elementary schools should adopt Scratch programming as part of their computer literacy course.

Wilson and Moffat (2010) conducted a research study to evaluate the use of Scratch in their IT courses for eight weeks, as an introduction to programming for total novices, in a younger age-group at primary school. The researchers focused on two dimensions of benefits; cognitive (if Scratch teaches concepts well), and affective (if it is fun to use) for the younger age-group in a school context. Their findings showed that benefits of both dimensions are important and they tend to feed back into each other which brought the researchers to the conclusion that: "… an ideal educational system to help learn how to

program should be designed with as much attention paid to the learner's emotional state as to the cognitive dimension" (p. 12).

In another study, Wilson, Hainey and Connolly (2013) also examined the use of game construction tools, namely Scratch, in the primary school classroom to familiarise children with programming. The researchers undertook an observational research and also presented a coding scheme for Scratch Games. The researchers concluded that overall, learners were able to make progress and had gained some programming concepts over the eight week period. Similarly, Baytak and Land (2011) conducted a case study to explore the processes used by 5th graders to design and develop computer games using Scratch which is based on constructionism approach.

The researchers concluded that "… elementary students can develop programming concepts and create computer games when using graphical programming software developed for their level of experience." (p.765).

From a different point of view, Calder (2010) examined the ways mathematical thinking emerges when children's work with an interactive, programming language, namely Scratch. The researcher required learners to design an activity and considered how this facilitated an authentic problem solving process. Calder (2010) concluded that "Scratch software proved to be an engaging and relatively easy-to-use space for problem solving, which at the same time provided a worthwhile and motivating programming environment to explore mathematical concepts" (p. 13). Hence, the educational benefits of Scratch environment for children are also obvious and teachers should consider these potential in their courses.

As a third and last step, there are also some more general recommendations provided by researchers. Brown and Kölling (2012) emphasised the importance of computing skills which can assist learners to read, understand and modify program code and algorithms to adapt to various situations. The researchers also stated their belief about the use of programming skills in the way that makes learners understand the phenomenon in a deeper and direct way. Using an eclectic approach, Smith (2010) reported on a tangible programming system designed for the novice user in developing regions. The system integrated three open source tools, namely Arduino, Processing and Scratch, and researchers stated that this attempt made the integration between hardware and software almost a trivial task. The proposed system was a simple one, having a low cost hand-made hardware incorporating recycled material and used to code a five-step sequence using an instruction set of four commands.

Realising the powerful dimensions of Scratch software like being a networked and media-rich programming environment, Maloney et. al. (2004) hypothesised that if learners work on individually meaningful Scratch projects, such as animated stories, games, and interactive art, they will develop technological fluency, mathematical and problem solving skills. Based on these ideas, researchers developed three short scenarios for possible implementation and made formal evaluations. From a different perspective, Scaffidi and Chambers (2012) have tried to explore the level of increase in users' skills over time in the Scratch environment. Their findings showed a positive progression of social skills whereas a negative progression of demonstrated technical expertise, as reflected in a wide range of programming primitives. Criticising their findings from different points

of view, the researchers called for future studies on this phenomenon. Moreover, from an instructional design point of view, Kordaki (2012) suggested a set of categories of learning activities that can be delivered to learners via Scratch. The researchers categorised these learning activities under 11 headings including possibilities such as creative activities, problem solving activities, working on the code for accomplishing different tasks, black-box activities, and collaborative learning activities.

Harvey and Mönig (2010) discussed in their paper whether or not Scratch can serve for two different target groups, namely for kids and advanced programmers. The researchers pointed out the growing trend among universities for implementing Scratch in computer science courses for beginners. In fact, the Scratch software is serving both groups since there are many researches dealing with different age groups who are also beginners in programming. This idea is also supported by Claypool (2013) who points out that game development is an effective approach for motivating learners to learn both beginner and advanced computer science topics. Resnick, et. al. (2009) also mentioned the use of Scratch from a wide of range of educational institutions from K–12 schools to universities as a first step into programming. Researchers mention successful implementations for early adapters but emphasise the need to provide better educational support for wider usage. They also underline the importance of a shift in perceptions about programming, and about computers in general. They concluded that "We need to expand the notion of 'digital fluency' to include designing and creating, not just browsing and interacting. Only then will initiatives like Scratch have a chance to live up to their full potential" (p. 67). Similarly, Romero (2010) also emphasised the appeal of programming tools for children and youth (in the scope of article tools for libraries) is that they help younger users develop 21st century skills such as problem solving, collaboration, and creativity.

There are numerous insights and hopeful findings about the use of Scratch in the process of teaching programming skills and concepts. Therefore, it seems that the use of Scratch for ages 8–18 will continue to evolve with these positive and meaningful findings for learners.

2. Method

2.1. *Research Design*

This study was carried out in explanatory sequential mixed methods design (Table 1). In this research design, researchers collected quantitative and qualitative data sequentially. "The rationale for this approach is that the quantitative data and results provide a general picture of the research problem; more analysis, specifically through qualitative data collection, is needed to refine, extend, or explain the general picture" (Creswell, 2012, p. 542).

The quantitative part was conducted in pre-test/post-test quasi-experimental design. The treatment variable of the study was Scratch programming. The dependent variable of the study was problem solving skills. The scores of the problem solving skills were gathered through Problem Solving Inventory (PSI). In the qualitative part of the research, students were observed by an independent observer with an observation form

Table 1
Design of the Study

Classes	Pre-test	Treatment (5 weeks)	Post-test
5 - A	PSI	T	PSI Focus group interview (FGW)
5 - B	PSI	T	PSI FGW
5 - C	PSI	T	PSI FGW

while they were programming. After the five-week experimental process, a focus group interview was conducted with students about experimental process with a structured interview form.

2.2. *Participants*

49 primary school students, who were attending a computer course at a private school in Turkey, participated in the study. This computer course is taught in one hour per week. Of the 49 students, 22 were female and 27 were male.

2.3. *Context and Process*

According to the new curriculum of the computer course in primary schools established in the 2012–2013 semester, 5th grade primary students have to be taught how to program a computer. In many countries, Scratch programming environment is being used to teach programming to children. In this context, an in-service distance training programme was organised for computer teachers in a private school. In this programme, teachers were taught theoretical and practical aspects of Scratch programming via video conferencing systems. Some of the topics were covered in this training programme are shown in Table 2.

The experimental process began with the application of PSI to students as a pre-test. Then students were taught topics such as an introduction to Scratch programming, installation of a Scratch platform, introduction to User Interfaces and the writing of programmes individually such as Hello World, Parrot, Aquarium programs and Maze project. Totally, the students have spent five hours to write these programs at school.

In the Hello World program, in contrast to the classic Hello World program, students added a background picture to their scenes. Then they assigned a meow sound command to the cat character, they moved the cat and finally they made the cat give messages such as “Welcome”, and “Today we will learn how to scratch” (Fig. 1).

In the Parrot program, students learned how to start the program, iterate the block and change the cat character. They made the parrot character fly by changing the costumes of the character (Fig. 2).

Table 2

Training programme of teachers

Topic
Introduction to Scratch programming Installation of Scratch platform Introduction to User Interface Hello World program
Working with blocks Parrot program
Working with blocks Aquarium program
Operators and Variables Maze program
Random Number and Jazz programs
What is your Name? program
Guess program

Fig. 1. Screen captures from Hello World program.

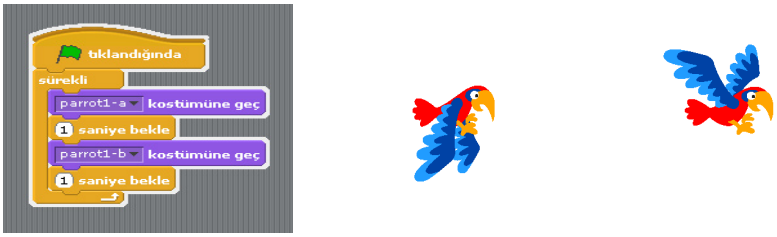

Fig. 2. Screen captures from Parrot program.

In the Aquarium program, students learned how to manage and control more than one character, and practiced how to iterate the blocks. Different from previous programs, they learned to how to turn the character if the character was on the edge, how to point towards the mouse cursor, how to change the colour effect, how to play sound until done (Fig. 3).

Students have spent two weeks on the Maze project by adding variables for calculating time spent in the game, life of the character and by adding sounds to the scene (Fig. 4).

Fig. 3. Screen captures from Aquarium program.

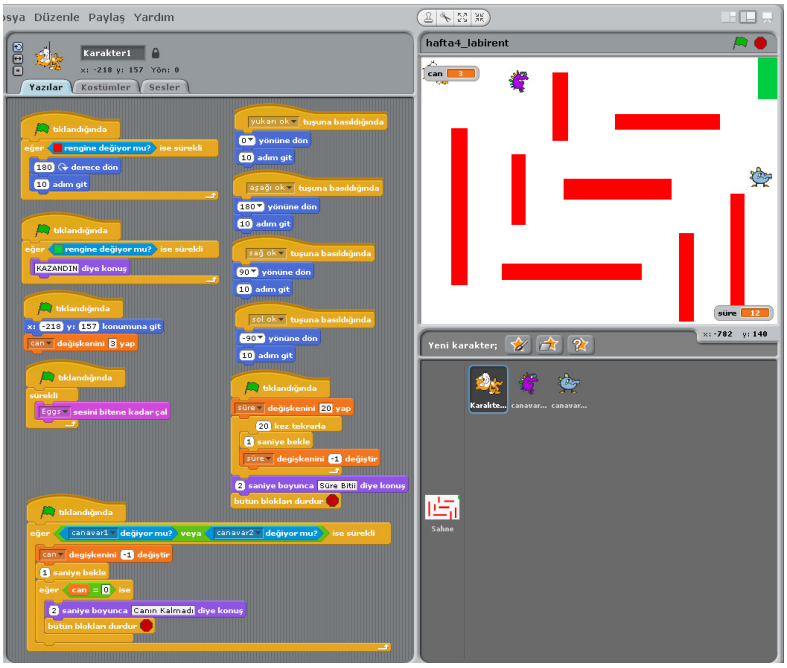

Fig. 4. Screen capture from Maze project.

Moreover, students learned how to use if conditions, how to use and activate arrow keys from the keyboard and how to change the X and Y positions of the characters.

All 5th grade students with different classes have completed the same programs with same teacher. An independent observer (she is also a computer teacher) observed a section every week with an observation form. After five weeks and all applications and projects were completed, PSI was administered to the students as a post-test.

2.4. *Research Questions*

The aim of this research study was to explore the effect of Scratch programming on 5th grade primary school students' problem solving skills. For this purpose, questions presented below were answered:

1. Does Scratch programming affect problem solving skills?
2. What do 5th grade primary school students think about programming?

2.5. *Data Collection Techniques*

2.5.1. *Quantitative – Problem Solving Inventory (PSI)*

As quantitative measure, Problem Solving Inventory of Serin, Bulut Serin, and Saygılı (2010) was used in this study. This measurement tool can be used for finding out the primary education school students' self-perception levels about their problem solving skills. After implementation of the inventory to 568 students from different grades (4th, 5th, 6th, 7th, and 8th) and after factor analysis, the Cronbach Alpha coefficient was found to be 0.80. In this inventory, there are 24 items with three factors; 12 items for self- confidence in their problem solving ability, 7 items for self-control and 5 items for avoidance.

2.5.2. *Qualitative – Focus Group Interview*

After completion of the programs and implementation of the inventory, teachers chose four or five students from each class according to their interest and success in Scratch programming. One of the researchers conducted a focus group interview with a structured interview form. An expert opinion about the questions used in the interview form was taken. According to the opinion of the expert, the expression of one item was corrected and the order of another was changed. The questions below 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. What was your favourite aspect of these applications?
d. What was your non-favourite aspect of these applications?
e. Was the use of platform easy? Did you face any problems within the platform?
f. Do you like programming?
g. What do you want to learn more about programming? Do you want to improve yourself?

Of the 13 students who participated to the focus group interview, 5 were female and 8 were male. Interviews were recorded with a voice recorder. The total duration of the interviews was 13:19, 06:59 and 7:51 minutes respectively.

2.5.3. *Qualitative – Observation*

While students were performing their tasks, an independent teacher observed the students according to a list of items that the researchers provided. An expert opinion about items in the observation form was also taken. The items were rated as 0, 1, 2 or 'not applicable'. The items below were used to observe each student in the class:

1. The student understands the program.
2. The student suggests different ways for writing the program.
3. The proposed solution of the student often gives the correct result.
4. The student can easily write programs.
5. The student is able to plan for the solution of the program.
6. The student is able to observe the output of the program.
7. The student is able to check the accuracy of the output of the program.
8. The student is easily able to solve problems encountered when writing a program.
9. The student asks to the teacher if there is a problem when writing a program.
10. The student asks to a friend if there is a problem when writing a program.
11. The student can write a similar program by himself/herself.
12. Anything else about student performance?

Every week, the observer could observe only one class because of her schedule. Thus, she observed class C, class A, class A, class C and class B respectively. Because of the time limitation of the study, class B could be observed once.

2.6. *Data Analysis*

To reveal whether there is a difference between pre-test and post-test, paired samples T-test was calculated with the data gathered from Problem Solving Inventory. 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. Finally, the items on the observation forms were summed up for each student and the least performed items identified.

3. Results

3.1. *Effect of Scratch Programming on Problem Solving Skills of the Students*

T-test results of pre-test and post-test of PSI are shown in Table 3. According to these results, there were no differences between pre- and post-test results of the students' problem solving skills [$t(48)=0.26$, $p>.05$]. For more detail analysis, T-test for pre-and post-test between sub-scales was also conducted. There were only non-significant increases in the mean of the factor of "self- confidence in their problem solving ability". In other words, programming in Scratch platform did not cause any significant differences in the

Table 3

T-test results of pre-test and post-test of PSI

Test	Mean	N	Std. Deviation	df	t	p
Pre-Test	2.07	49	0.52	48	0.026	0.980
Post-Test	2.07	49	0.58			
Pre-Test of Self- Confidence in Their Problem Solving Ability	2.10	49	0.64	48	-0.670	0.506
Post-Test of Self- Confidence in Their Problem Solving Ability	2.17	49	0.74			
Pre-Test of Self-control	2.06	49	0.68	48	1.491	0.143
Post-Test of Self-control	1.94	49	0.74			
Pre-Test of Avoidance	2.03	49	0.61	48	0.305	0.762
Post-Test of Avoidance	2.01	49	0.70			

problem solving skills of the primary school students. This result may show that programming in Scratch platform may not have an impact on the problem solving skills.

As can be seen from the results of the quantitative data, however, the students' self-perception about their problem-solving skills was found to be very low. Students have to be supported with different activities that require high-order thinking in order to help them develop problem solving skills.

3.2. *Thoughts of the Students about Programming*

3.2.1. *Difficulties about Problem Solving Process*

When students were asked whether or not they had difficulties regarding the problem solving process, 7 students said that they had difficulty. One student said that the Maze project was difficult to handle. To solve this situation, 5 students said that they got help from their teacher, while 2 explained that they got help from a friend (Table 4). One student expressed his opinion as "Obviously, I sometimes had difficulties. But the program

Table 4

Difficulties about problem solving process

Themes	#
Have difficulty in problem solving process/Sometimes have difficulty in problem solving process	7
Reasons	
Maze project was difficult	1
Solutions	
Get help from the teacher	5
Get help from a friend	2
Don't have difficulty in problem solving process	6
Reasons	
Follow the teacher's instructions	5

was so nicely planned, that is to say, that you can see where everything is when looking around the program". Unlike students with problems, 6 students said that they did not have any difficulty with the problem solving process. 5 students said that the process was easy because they carefully followed the instructions of the teacher.

3.2.2. *Opinions about Solving Problems in Different Ways*

When students were questioned about solving problems in different ways, 7 students said that they solved their problems in different ways. In this context, 2 students said that they added a monster to their program, while 2 students added a bounce command to the cat character (Table 5). While one student said "Incidentally, I've changed something, and then started to make changes in a more conscious way", another student explained "Once, I was playing around with the program and everything was reset; I had to start again from the beginning". Unlike students adding new things to their program, 6 students explained that they did not solve their problem in different ways. In this sense, 3 students said that they did what the teacher told them.

3.2.3. *Favourite aspect of these applications*

For the question regarding their favourite aspects of these applications, 7 students expressed that they liked to give a command to a character, 5 students liked to learn how to write a game, 3 students liked to add a variable to the programs, 2 students liked to share what they did with others and finally 1 student liked to create his own world (Table 6).

Some expressions of the students such as "Like us, they [the characters] were moving, and we were selecting this.", "I thought it was a terrible thing to tell someone what to do,

Table 5
Opinions about solving problems in different ways

Themes	#
New ways	7
Add a monster	2
Add bounce command	2
No different ways	6
Did what the teacher said	3

Table 6
Favourite aspect of these applications

Themes	#
Give a command to a character	7
Learn how to write a game	5
Add variable to the programs	3
Share what I do with others	2
Create own world	1

but actually, it was nice", "When playing games on the computer, normally I was very curious how they were done. I thought it was going to be difficult, but it actually came easy to me when I started to practice Scratch", "I liked to revive a lifeless thing".

3.2.4. *Non-favourite Aspect of these Applications*

On the responses of students regarding their non-favourite aspects of these applications, 4 students mentioned that there were no special effects that could be applied to the characters, they found Scratch very simple. Deletion of blocks (3), the complexity of long blocks (2), not finding the desired commands (1) and limited characters and scenes were other non-favourite aspects that the students stated (Table 7).

3.2.5. *Ease of Use of Platform and Problems with the Platform*

When students were asked about ease of use of platform and problems faced with the platform, 10 students stated that the platform was easy to use. Two students' opinions on this issue were "First class students who can read and write can even use it" and "It is already separated by colour. Very simple". 4 students said that they did not understand the variables, 3 students pointed out that the platform was hard to use at first, but easy once they learned how. 4 students mentioned problems. 2 of them faced problems recording and adding audio and others 2 stated problems about editing and moving the characters (Table 8).

Table 7

Non-favourite aspect of these applications

Themes	#
No special effects / very simple	4
Deletion of blocks	3
Difficult to draw the desired character	2
The complexity of long blocks	2
No non-favourite aspects	1
Not finding the desired commands	1
Limited characters and scenes	1

Table 8

Ease of use of platform and problems with the platform

Themes	#
Ease of use of platform	
Easy	10
Not understanding variables	4
Hard at first, but easy once learned	3
Problems	
Recording and adding audio	2
Characters	2

3.2.6. *Thoughts about Programming*

When students were asked whether they liked programming or not, all students stated that they liked programming and 3 of them also explained that programming is very entertaining. Students' opinions on this subject were:

"… it is a very nice feeling when I did something."

"… when you made a game [and] when it is played a lot, you're very happy."

"Very nice for the children to develop their brains."

"To give life to an inanimate thing ... It is like creation of Frankenstein by a doctor."

3.2.7. *Their Opinions on their Future Programming*

When students were questioned about what they want to learn from programming, all the students stated that they want to improve themselves in programming. Moreover, 6 of them stated that they want to create larger, more difficult and 3D games. 2 of them expressed that they want to learn Photoshop and two others want to write program code (Table 9). In this sense, a student stated that "We write program with blocksnstead of this, computer programmers are using just some of the letters, as it [Scratch] shows you a simple way to do it." and "I think Scratch application is very reasonable because it is very nice, fun and easy. This is intended for us because children are curious about the games us children like to use. It's really beautiful ... "

3.3. *Results of Observation Process*

When results of the observation process were examined after five weeks, the students' performance on programming was found to be similar (Table 10). While Class 5-C got

Table 9

Opinions on their future programming

Themes	#
Want to improve themselves in programming	13
Create a larger / more difficult / 3D game	6
Learn Photoshop	2
Want to write program code	2

Table 10

Points range gathered from observation form

Observation Week	Class	Points range gathered from observation form
Week 1	5-C	15-22
Week 2	5-A	12-22
Week 3	5-A	18-22
Week 4	5-C	15-22
Week 5	5-B	18-22

the same points after two weeks, Class 5-A got higher points after a week. Considering that Class 5-B was only observed once, they showed a good performance.

Considering that if all the items were fully performed, a maximum of 22 points can be achieved; students can be characterised as having shown a pretty good performance. When the most poorly performed items were analysed, item 2 (The student suggests different ways for writing the program), item 3 (Proposed solution of the student, often gives the correct result), item 10 (The student asks to a friend if there is a problem when writing program) and item 11 (The student can write a similar program by himself/herself) were found to be the least observed items. In fact, these results are not surprising, because students stated in the focus group interviews that a few of the students solved their problems in different ways and very few students asked a friend if there is a problem when writing their programs.

4. Discussion and Conclusion

In this study, the researchers explored the effect of Scratch programming on 5th grade primary school students' problem solving skills. Moreover, they wondered what 5th grade primary school students think about programming. According to the quantitative results, programming in Scratch platform did not cause any significant differences in the problem solving skills of the primary school students. This result may show that programming in Scratch platform may not have an impact on their problem solving skills. This result can only be considered within the scope of this research. Different results can be achieved in a different context and research designs. However, although this study was carried out over a short time period and with a small number of applications; there was a slight improvement in the students' self-confidence in their problem-solving ability. This slight improvement is also valuable for these conditions. In fact, this sheds light on the possibility that programming may affect problem-solving skills of the students.

Another point to be considered is that the students' self-perception about their problem-solving skills was found to be very low. Students have to be supported with different activities and applications that require high-order thinking in order to help students develop problem solving skills.

When the thoughts of the primary students were considered, it can be clearly said that students liked programming and wanted to improve their programming. In the problem solving process, while half of the students had some difficulty, the others did not. Most of them tried to solve their problems in different ways. The favourite aspect of these applications were assigning a command to a character, learning how to write a game, adding a variable to a program, sharing what they did with others and creating their own world. The least favourite aspects were that no special effects can be applied to the characters, the deletion of blocks, the complexity of long blocks, not finding the desired commands and limited characters and scenes. Finally, most of the students found the platform easy to use.

Based on the literature and expert opinions, it is obvious that computer programming can enhance problem solving capabilities of learners in all ages, which urges a need for

the development of educational programming environment based multimedia activities, especially for young learners. On the other hand, just providing the learning environment will not be sufficient to fulfil the need of effective teaching. Hence, as also stated by (Fessakis, Gouli and Mavroudi, 2013) "Experimentally validated teaching/learning approaches, documented best practices, learning resources, curriculum standards, professional development and support for teachers are also needed (p. 96).

Acknowledgements

Thanks to computer teachers, Sümeyra Akçay and Ebru Dere for their help and assistance in this research.

References

Baytak, A., Land, S. (2011). An investigation of the artifacts and process of constructing computers games about environmental science in a fifth grade classroom. *Educational Technology Research and Development*, 59(7), 765–782.

Brown, N., Kölling, M. (2012). Position paper: programming can deepen understanding across disciplines [DRAFT]. In: *IFIP Working Conference – Addressing Educational Challenges: the Role of ICT. Manchester Metropolitan University, July 2 – 5, 2012*. Manchester, UK. http://www.cs.kent.ac.uk/people/staff/nccb/position_paper.pdf

Calder, N. (2010). Using Scratch: an integrated problem-solving approach to mathematical thinking. *Australian Primary Mathematics Classroom*, 15(4), 9–14.

Cegielski, C.G., Hall, D.J. (2006). What makes a good programmer? *Communications of the ACM*, 49(10), 73–75.

Claypool, M. (2013). Dragonfly: strengthening programming skills by building a game engine from Scratch, *Computer Science Education*, 23(2), 112–137.

Creswell, J.W. (2012). *Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research* (4th Ed.). Boston: Pearson Education.

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.

Harvey, B., Mönig, J. (2010). Bringing "No ceiling" to Scratch: can one language serve kids and computer scientists?. *Constructionism 2010, Paris*, 1–10. http://byob.berkeley.edu/BYOB.pdf

Kim, H., Choi, H., Han, J., So, H. (2012). Enhancing teachers' ICT capacity for the 21st century learning environment: three cases of teacher education in Korea. *Australasian Journal of Educational Technology (AJET)*, 28(6), 965–982.

Kordaki, M. (2012). Diverse categories of programming learning activities could be performed within Scratch. *Procedia – Social and Behavioral Sciences*, 46, 1162 – 1166.

Lai, A., Yang, S. (2011). The learning effect of visualized programming learning on 6th graders' problem solving and logical reasoning abilities. In: *International Conference on Electrical and Control Engineering (ICECE), 16–18 Sept. 2011, Yichang,* 6940–6944.

Lamb, A., Johnson, L. (2011). Scratch: computer programming for 21st century learners. *Teacher Librarian*, 38(4), 64–68.

Lee, Y. (2011). Scratch: multimedia programming environment for young gifted learners. *Gifted Child Today*, 34(2), 26–31.

Lewis, C. M. (2011). Is pair programming more effective than other forms of collaboration for young students?. *Computer Science Education*, 21(2), 105–134.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., Resnick, M. (2004). Scratch: a sneak preview. In: *Second International Conference on Creating, Connecting, and Collaborating through Computing*. Kyoto, Japan, 104–109.

Nelson, J. (2009). Celebrating Scratch in libraries: creation software helps young people develop 21st-century literacy skills. *School Library Journal*, 20–21.

Ornelas Marques, F., Marques, M.T. (2012). No problem? No research, little learning ... big problem!. *Systemics, Cybernetics and Informatics*, 10(3), 60–62.

Papadopoulos, Y., Tegos, S. (2012). Using microworlds to introduce programming to novices. In: *Proceeding PCI '12: Proceedings of the 2012 16th Panhellenic Conference on Informatics*, Piraeus, Greece, 180–185.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E, Siver, J., Silverman, B., Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60–67.

Romero, J.S. (2010). Library programming with lego mind storms, Scratch, and picocricket: analysis of best practices for public libraries. *Computers in Libraries*, 30(1), 16–45.

Scaffidi, C., Chambers, C. (2012). Skill progression demonstrated by users in the Scratch animation environment. *International Journal of Human-Computer Interaction*, 28(6), 383–398.

Serin, O., Bulut Serin, N., Saygılı, G. (2010). İlköğretim düzeyindeki çocuklar için problem çözme envanteri'nin (ÇPÇE) geliştirilmesi. *İlköğretim Online*, 9 (2), 446–458.

Smith, A. C. (2010). Dialando: tangible programming for the novice with Scratch, processing and arduino. In: *6th International Workshop on Technology for Innovation and Education in Developing Countries, Maputo, Mozambique, 21–23 January 2010*, 1–4.

Theodorou, C., Kordaki, M. (2010). Super Mario: a collaborative game for the learning of variables in programming. *International Journal of Academic Research*, 2(4), 111–118.

Wang, X., Zhou, Z. (2011). The research of situational teaching mode of programming in high school with Scratch. In: *6th IEEE Joint International Information Technology and Artificial Intelligence Conference (ITAIC), 2011, Chongqing*, 488–492.

Wilson, A., Moffat, D. C. (2010). *Evaluating Scratch to introduce younger schoolchildren to programming*, 1–12. `http://scratched.media.mit.edu/sites/default/files/wilson-moffat-ppig2010-final.pdf`

Wilson, A., Hainey, T., Connolly, T.M. (2013). Using Scratch with primary school children: an evaluation of games constructed to gauge understanding of programming concepts. *International Journal of Game-Based Learning*, 3(1), 93–109.

Xinogalos, S. (2012). An evaluation of knowledge transfer from microworld programming to conventional programming. *Journal of Educational Computing Research*, 47(3) 251–277.

F. Kalelioğlu is working at Baskent University as the faculty member of Department of Computer Education and Instructional Technology. Her main research interests are e-learning, social networks, instructional design and technology integration.

Y. Gülbahar is working at Ankara University as the head of Department of Informatics. Her main research interests are e-learning, mobile learning, programming and algorithms, instructional design, computer education and educational technologies.

Programavimo mokymo naudojant Scratch efektyvumas: mokinio perspektyva

Filiz KALELIOĞLU, Yasemin GÜLBAHAR

Programavimas yra svarbi kompetencija problemų sprendimo įgūdžiams formuoti ir loginiam mąstymui ugdyti. Todėl programavimo integravimas visuose švietimo lygmenyse, taip pat ir ankstyvajame amžiuje, yra reikšmingas ir šioje srityje atlikta nemažai tyrimų. Pristatomas tyrimas, kurio tikslas – nustatyti programavimo mokymo naudojantis Scratch efektyvumą pradinės mokyklos penktos klasės mokiniams, siekiant ugdyti jų problemų sprendimo įgūdžius. Be to, buvo siekiama išsiaiškinti, ką penktokai galvoja apie programavimą. Tyrime dalyvavo 49 pradinės mokyklos mokiniai. Remiantis kiekybinių rezultatų analize, programavimas naudojant Scratch nepateikė jokių reikšmingų skirtumų ugdant pradinės mokyklos mokinių problemų sprendimo įgūdžius. Pastebėtas tik vienas nereikšmingas vidurkio padidėjimas „pasitikėjimo spręsti problemas“ faktoriaus atžvilgiu. Analizuojant mokinių samprotavimus apie programavimą, vienareikšmiškai nustatyta, kad jiems patiko programuoti ir jie norėtų pagerinti programavimo įgūdžius. Tyrime dalyvavę mokiniai teigė, kad Scratch platforma yra lengvai naudojama.