# Plugged or Unplugged Teaching: A Case Study of Students' Preferences for the Teaching Programming

**Ali Kürşat Erümit**[1]
**Güven Şahin**[2]

[1]Trabzon University
[2]Recep Tayyip Erdoğan University

DOI: 10.21585/ijcses.v4i1.82

**Abstract**

This study is an investigation of the effects of plugged and unplugged activities in a programming course using the Programming in Seven Steps (PSS) model on pupils' satisfaction and activity type preferences. A case study method was used in the classroom was the case. Data included students' diary, their responses on semi-structured interview forms, and semi-structured interviews with a selected sub-set of students. The collected data were analyzed by content analysis technique. It has been found that there are different factors that positively affect student satisfaction for "Conditional Structures", "Variables" and "Loops" courses that are processed according to YAP model. In addition, students' preferences and reasons for activity type differ. Study results show that training students with their preferred activity types increases their satisfaction and enable them to overcome associated difficulties more easily. It is concluded that because the PSS model is effective with students with different learning environment preferences, it can be used as a model to increase learner satisfaction with programming instruction.

**Keywords:** teaching programming, PSS model, plugged activities, unplugged activities, student satisfaction

## 1. Introduction

With the rapid advance of the developments in the field of science and technology since the industrial revolution up to the present, the skills individuals need to have in the 21st century have evolved. Wagner (2008) lists such skills as critical thinking and problem solving, quick wit and adaptation, entrepreneurship and taking initiative, and inter-system and interpersonal cooperation as well as leadership, effective oral and written communication, curiosity and imagination, and ability to access and analyze information.

Studies on acquisition of these 21st century skills reveal that teaching programming significantly improves students' logical thinking (Shih, 2014), problem solving (Brown et al., 2013; Lai and Lai, 2012; Kalelioğlu and Gülbahar, 2014), computational thinking (Sáez-López et al., 2016), reflective thinking (Fesakis and Serafeim, 2009; Kobsiripat, 2015), metacognition (Fessakis, Gouli, and Mavrodi, 2013; Kafai and Burke, 2014), and collaborative work (Denner, Werner, and Ortiz, 2012). In addition to these skills, learning programming has been found to have a positive effect on students' academic achievement in a variety of subjects (computers, mathematics, science, language arts, and arts education) (Calder, 2010; Sáez-López et al., 2016).

The literature shows that teaching programming holds an important potential for helping students gain 21st century skills. Through programming teaching in the literature; students' logical thinking (Lai & Lai, 2012; Shih, 2014), problem solving (Brown et al., 2013; Lai & Lai, 2012), computational thinking (Sáez-López et al., 2016), higher level thinking There has been positive progress in skills (Kafai & Burke, 2014) and critical thinking (Erümit et al.,

2019). Besides developing students' cognitive skills, however, programming requires other higher level skills for the learning process itself (Law, Lee and Yu, 2010), which makes learning programming is quite challenging (Helminen and Malmi, 2010), and students have an overall lower level of achievement in this subject (Robins, Rountree, and Rountree, 2003). In order to increase the success of teaching programming and to facilitate students' understanding, it is necessary to first teach the logic of algorithms to students (Ala-Mutka, 2004). For this purpose, interesting and entertaining visual programming tools have been developed in order to facilitate learning for beginners in programming education (Schwartz, Stagner, and Morrison, 2006).

Teaching programming entails challenges in selecting suitable activities for a group of learners (Çatlak, Tekdal, and Baz, 2015), helping students understand and apply algorithms (Futschek and Moschitz, 2010), and helping them use programming languages in writing codes (Arabacıoğlu, Bülbül, and Filiz, 2007). The syntax of text-based programming is considered as one of the most challenging issues for students (Mannila, Peltomaki, and Salakoski, 2006; Özmen and Altun, 2014), while the code blocks presented in visual programming are easier for beginning level programmers to understand and apply (Wilson and Moffat, 2010). Calder (2010) argues that working with the block-based visual tools of programming software increases students' satisfaction and motivation to persist. In many studies, it was noted that the use of visual programming accelerates comprehension of the process (Naharro-Berrocal, Pareja-Flores, Urquiza-Fuentes, and Velazquez-Iturbide, 2002). For beginning learners, therefore, elementary level software programs such as Scratch (Malan and Leitner, 2007; Wu, Chang and He, 2010), Kodu (Stolee and Fristoe, 2011), StarLogo (Klopfer and Yoon, 2005), and Alice (Kelleher, Pausch, and Kiesler, 2007) are recommended as they allow students to perform coding by placing code blocks in order through drag and drop functions. In this way, the frequently faced problem of syntactic errors in text-based programming can be largely overcome.

In addition, attitudes toward programming (Gomes and Mendes, 2007; Hawi, 2010) and self-efficacy beliefs (Aşkar and Davenport, 2009) are some of the affective issues encountered in the teaching programming. These challenges often arise during the teaching of basic programming concepts such as the structure of a programming language (Lahtinen, Ala-Mutka, and Järvinen, 2005), loops (Ginat, 2004), and algorithm structures (Seppälä, Malmi, and Korhonen, 2006).

Another major challenge encountered in teaching programming relates to overall pedagogical approaches (Coull and Duncan, 2011; Lahtinen, Mutka, and Jarvinen, 2005), such as the following as found in the literature: Problem solving (Asad, Tibi, and Raiyn, 2016; Grover, Pea, and Cooper, 2016; Qian and Lehman, 2016), abstraction (Futschek and Moschitz, 2010), peer instruction (Denner, Werner, and Ortiz, 2012), collaborative learning (Denner, Werner, and Ortiz, 2012; Denner et al., 2014), writing algorithms (Futschek, 2006; Futschek and Moschitz, 2010; Milková and Hůlková, 2013), and drama or role plays (Karaosmanoğlu and Adıgüzel, 2017; Sarıoğlu and Kartal, 2017; Weigend, 2014).

In the implementation of these approaches, both plugged and unplugged activities (activities with and without use of computers) may be used. For example, Sáez-López, Román-González and Vázquez-Cano (2016) carried out a study with 107 middle schoolers using Scratch and computerized activities. Their purpose was to explore the differences, if any, their approach made in the students' attitudes and competencies. They found that the students made progress in satisfaction, attachment, computational thinking, and calculating skills. Taylor, Harlow and Forret (2010) implemented plugged activities with 9- and 10-year-old students using Scratch and found out that the learners were interested in Scratch as a programming tool. In addition, they pointed out that the tool provides an environment in which students employ problem solving processes such as idea generation, target setting and testing. Similar studies using visual programming tools like Scratch have shown that such tools improved students' satisfaction, interest, and enjoyment in learning (Kalelioğlu and Gülbahar, 2014; Kelleher et al., 2007; Malan and Leitner, 2007; Pinto and Escudeiro, 2014; Wu et al., 2010).

Especially at an early age, there are difficulties in learning abstract programming topics. As a solution, unplugged activities can be used to teach programming. Because unplugged activities contain the lowest level of technical information, they involve fewer cognitive challenges than other programming tasks and are commonly used in basic computer instruction as a starting point before student progress to plugged activities (Kotsopoulos et al., 2017). Starting teaching programming according to the activity type preferences (plugged/unplugged) of the students or planning the activities to include both types of activities can facilitate the teaching programming for younger students and affect their perceptions and attitudes towards programming positively (Şahin, 2018). For example, Futschek and Moschitz (2011) introduced unplugged activities prior to plugged activities to students at the basic level of programming. They found out that this sequence of teaching increased students' satisfaction with plugged activities, and even those who previously had not been interested in programming ended up being

enthusiastic. Giannakos et al. (2013) also found that using both plugged and unplugged activities increased students' interest in and satisfaction with both programming and computer sciences as a subject. They also found that female students became more interested in computer sciences than their male peers. Hermans and Aivaloglou (2017) investigated which method, plugged or unplugged, is more effective as a starting activity in programming in order to facilitate learning the concepts of programming, support learners' self-efficacy in programming tasks, and motivate them to do research. As a result, the group starting with unplugged activities recorded higher self-efficacy. Also, those students used a larger range of Scratch blocks.

Pedagogical approaches and types of activities (plugged and unplugged) mentioned in the literature, however, tend to be applied independently in programming instruction. Moreover, there exists no specific model for these coordinating these pedagogical approaches in class and laboratory settings with the age of the target group, for making them workable steps, and thus for systematizing programming instruction. Targeting these problems, Erümit et al. (2019) devised a teaching model entitled Programming in Seven Steps (PSS) for teaching programming so as to improve students' algorithmic thinking and problem-solving skills independent of any specific programming environment.

The seven steps of the model include plugged and unplugged activities as shown in Figure 1:

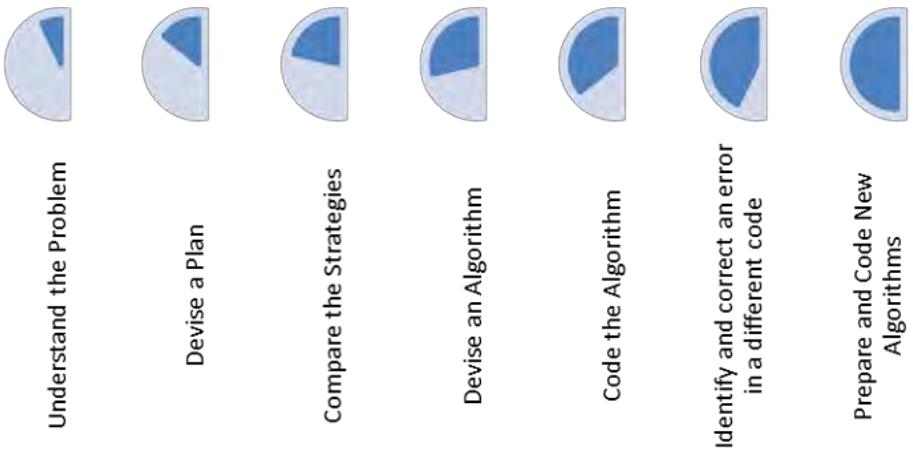



Figure 1. Steps of the PSS Model

The first four steps, "Understand the Problem," "Devise a Plan," "Compare the Strategies," and "Devise an Algorithm," use unplugged activities. The last three steps, "Code the Algorithm," "Identify and Correct an Error in a Different Code," and "Prepare and Code New Algorithms," involve plugged activities. These steps are further elaborated below.

The first step - understand the problem involves identifying what relevant information is given in the problem, what further information is needed, and what given information is irrelevant, more or less simultaneously. The next step, devise a plan, requires developing strategies appropriate to seeking the solution of the problem and deciding how to implement them. In the step comparing the strategies, the results of implementing different strategies are evaluated and compared to determine through negotiation the best path to the solution. In the following step, devise an algorithm, the most effective algorithm emerging from the previous step is determined, and in the following step, the generated algorithm is coded. Identify and correct an error is a debugging process independent of the other steps to detect errors in the coding of a program. Finally, prepare and code new algorithms, involves revising the algorithm and/or generating new algorithms suitable for the given problem and coding them.

The purpose of the current study was to discover students' preferences for plugged and unplugged activities along with their justifications and also their satisfaction with the activities performed in the programming lessons taught with the PSS model. For this purpose, answer was sought for the following questions:

1) What is the effect of applying the PSS model in programming lessons l on students' satisfaction with the instruction provided?

2) What type(s) of activities do students prefer in programming lessons employing the PSS model and what are the reasons for their preferences?

## 2. Method

### *2.1. Research Model*

Case study was the qualitative research method used in this study. Qualitative research is a research model which uses non-metric data collection methods such as observations, interviews, and document analysis, and follows procedures for eliciting perceptions and depicting events in a realistic and holistic way in their natural setting (Yıldırım and Şimşek, 2005). It also provides flexibility to the researcher during the design and execution of the research (Silverman, 2013). Among many qualitative research designs, McMillan (2000) describes case study as a method for in-depth exploration of one or more events, settings, programs, social groups, or other interconnected systems. It is a research pattern in which an entity is defined and customized with reference to time and space (Büyüköztürk, Çakmak, Akgün, Karadeniz, and Demirel, 2016). To allow in-depth investigation of different aspects of a particular case, a variety of data sources and collection techniques are used (Cohen, Manion, and Morrison, 2005).

In this research, case study was chosen to achieve the following:

- in-depth content analysis of the data collected from the students related to activities offered by the PSS model,
- better understanding feelings, thoughts and emotions through qualitative methods,
- evaluating the findings in a descriptive way.

### *2.2. Participants*

Participants were 38 sixth grade students (20 girls and 18 boys) attending a public school, when the lesson plans were used in classes. Students were coded from K1 to K38. Students study in a classroom with average achievement in a school with a medium ranking among schools with computer lab. They had not previously used Scratch or received any programming training. The research-related activities were carried out by the teacher in charge of Information Technologies and Software (ITS) course, who had 15 years of professional experience and 5 years of experience in teaching Scratch programming. The researcher personally collected data while visiting the school throughout the implementation stage.

### *2.3. Data Collection Tools*

Data included students' journals (Appendix B), periodic semi-structured interviews responses with a sub-set of students, and their responses to a semi-structured interview form (Appendix A), developed by the researcher to explore students' perceptions and evaluation of the process. While preparing the data collection tools, the researcher consulted with six field specialists (two lecturers in Computer Education and Instructional Technologies, two PhD candidates, and two master's students). Thus, the scope validity of the data collection tools was ensured. Data collection tools were applied to 10 different students (5 females, 5 males) in the sixth grade before the actual application and their opinions were obtained and the necessary arrangements were made, and their opinions were obtained again. In this way, the validity of meaning is also provided.

### *2.4. Data Collection and Analysis*

All students wrote journal entries every week at the end of each week (nine times in total, see Appendix B). Also, the researcher and the teacher jointly conducted five semi-structured interviews with nine students (each student individually) at the juncture between one topic and the next. Interviewees were selected to represent a balance of mathematics and ITS proficiency levels (three low, three medium, three high) and genders (five females and four males). Interviews were audio-recorded, and all students' data were transferred to the computer for content analysis. Participants' informed consent was obtained prior to the interviews, and they participated on an absolutely voluntary basis. All participants' information was kept confidential.

*2.5. Validity and Reliability of Study*

The qualitative data were analyzed using content analysis methods, which included combining, reducing and interpreting what the participants said and what the researcher witnessed and read (Merriam and Tisdell, 2015). Audio-recordings of the student interviews were recorded, digitalized, and analyzed which included editing, coding by themes, and supporting the themes with citations and excerpts (Creswell, 2013). In addition, during the interviews the researcher kept detailed notes, paying due diligence to capturing all information. All data were securely saved in case of need to verify the results. Three researchers specializing in the field of Computer Education and Instructional Technologies first analyzed the data independently and then aligned their results. Initially the researchers' analyses were 88% congruent, following which they convened and reached full consensus on the study themes and related areas. The data were confirmed by member-checks by participants when deemed necessary. The researchers played an active role in reporting as well as in analyzing data.

*2.6. Implementation Approvals*

Because the research involved human subjects and their data, an approval certificate was obtained from the Ethics Committee for Social and Human Sciences of University, documenting conformity with ethical principles of purpose, justification, method and data collection instruments and procedures. The document was then submitted to the Ministry of National Education (MoNE), which granted approval for research in schools. After both approvals were obtained, an information form about the study and consent form were sent to parents of the participant students. The study was conducted with students only after receiving their parents' consent.

*2.7. The Implementation*

The research was carried out for 9 weeks (80 minutes each) with 38 students studying in the 6th grade. Each of the students in the class had his/her own computer, which was controlled from a central computer. In the first week, a researcher informed the students about the implementation process and its importance. Within the scope of the seven-step lesson plans, the first four steps were carried out without computers, while the other three steps were plugged. The weekly implementation process is shown in Figure 2.

7 steps of PSS model may not be applied in the same week. This is related to the duration of the lesson. The important thing is to use 7 steps to teach the same subject. In the same week, 7 steps of PSS may not catch up since the lesson time is limited. However, since each step constitutes a meaningful piece, it is not a problem that 7 steps are independent.

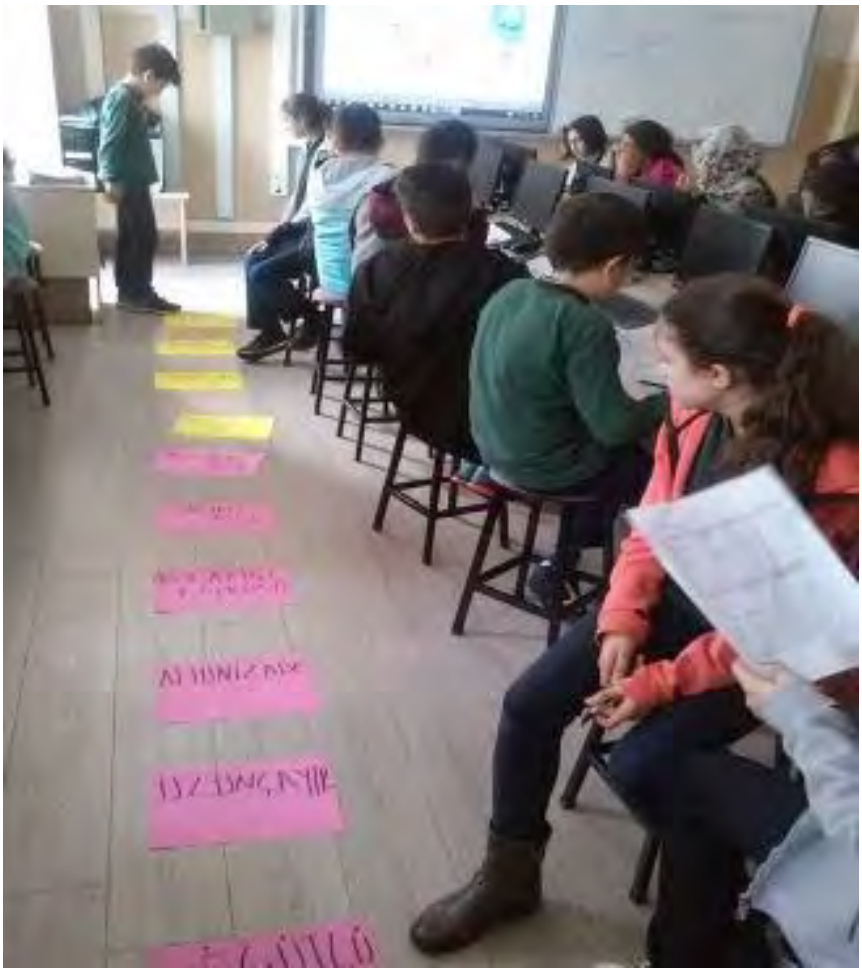
As shown in Figure 2, the first week covered Problem-Solving Concepts and Approaches. In the first lesson, the activity, "Helping the Shepherd" (Appendix C) was carried out with an activity sheet providing the scenario of the problem, a story visual, and a space in which to write the solution algorithm using pseudo-code. In the following lesson, the activity "I am finding the cities" (Appendix D) also included an activity sheet for students to analyze and solve a problem. Again, they used the blank area to provide the solution algorithm created through pseudo-code.

During the two classes of week two, the topic "Let's Recognize Scratch" was addressed. In the first lesson, the students studied sample projects on the Scratch web platform, guided by the course teacher to make sure they paid attention to the salient features of the projects. In order to ensure that the students pay attention to the characteristics of the projects they are studying, instructions were given by the course teacher (Appendix E). The next lesson was allocated for students to freely perform activities inspired from the projects examined. In the following week, the previous week's topic was continued. In the first lesson, the students watched Scratch training videos via the Education Informatics Network (EBA, http://www.eba.gov.tr/). In the next class by students completed some easy exercises.


Figure 2. The Implementation Processes

In week four the "Conditionals" were taught, again in two lessons beginning with the activity, "Ali and Ayşe are going on holiday," (Appendix F) in which the first four steps of the PSS model were applied. For "Understand the problem", the students answered questions about the problem situation using the Problem Acknowledgement Table (PAT) form (Appendix G). For "Devise a Plan", they used the Strategy Devise Form (SDF) (Appendix H) to decide on the most appropriate strategy and solved the problem.

For the following step, "Compare the Strategies," the teacher worked with the students to identify the most appropriate strategy from those generated in the previous step. As the next step, the students wrote the algorithm for their own plan using pseudo-code. Later, some students voluntarily dramatized their algorithm using materials provided by the teacher (Picture 1).



Picture 1. In Class Application

The next lesson began with a Scratch computer application, “Questions and Answers,” followed by “Maze Map” as the fifth and last step of the model in this phase.

Picture 2. “Maze Map” Scratch Activity

In the fifth week, the topic from the previous week was continued. In lesson one, the teacher presented a Scratch application with an incorrect code structure which the students studied to detect and correct the errors.

Incorrect Correct


Figure 3. Incorrect and Correct Code

During the next class, the students and the teacher finished the activity by modifying the faulty code structure together. After that, the teacher sent the students an assignment via the EBA system, which corresponds to steps 6 and 7 of the PSS Model.

In the sixth week, the topic of Variables was discussed. The first hour was used for an activity titled "Ali needs fruit," in which students completed the first four steps of the model. The next lesson followed for the implementation of Step 5 of the PSS Model in the context of "Fruit Basket" as a Scratch application. The seventh week was a continuation of the previous week. The procedure was put into practice as described under "Conditionals" in the second week

During week eight, "Loops" was discussed. The first lesson, which featured the activity "Ali playing a game," was completed after the first four steps of the PSS model as described in the fourth week had been implemented. The following class started with the Scratch computer application "My fuel amount," followed by "Auto racing," finalizing stage 5 of the PSS model. In the ninth week of the implementation, the issue of "Loops" continued. The application in the ninth week was carried out in the same way as the application in the fifth week, and the whole application process was completed.

To recapitulate the scope of the PSS model, as part of the first step, understand the problem, the students answered the questions formed in accordance with the activity in the lesson plan. In this step, the students were asked to distinguish the necessary and unnecessary information in the problem presentation. During the second step, devise a plan, the students sought the best among multiple paths to reach a solution to a problem. In step three, comparing the strategies, the teacher constructed the optimum result by examining all the students' proposed solutions together. Step four, code the algorithm, was completed in two parts. First, the students represented their solutions as algorithms written in pseudo-code. Later, volunteers acted out their algorithms in dramatic performances for the class. All of these four steps could fit into one class hour. During the next step, code the algorithm, the students undertook the task of transferring the previous activity onto Scratch. This step took about two hours.

In step six, identify and correct an error in a different code, the students were given Scratch applications with missing or incorrect code structures independent from the activity in the lesson plan and required to identify and eliminate the errors. After they completed the task individually, the teacher collaborated with the students to complete it on the smart board. This stage took one lesson period.

At the final stage, prepare and code new algorithms, the teacher mailed assignments to the students via EBA with specified criteria. In this assignment, the students were asked to first write the algorithm that met the criteria given and then to encode the algorithm in the programming environment. An overview and use of lesson plans are given in Table 1.

Table 1. Summary Table for Lesson Plans

| Topic | Attainment | Activity | PSS Step | Type | Individual or Group | Duration | Week |
|---|---|---|---|---|---|---|---|
| Problem Solving Concepts and Approaches | • Analyzes a problem.<br>• Describes the importance of generating authentic solutions for problems.<br>• Devises an algorithm for solving the problem.<br>• Discusses the problem-solving process and basic concepts. | Helping the Shepherd | - | Unplugged | Individual or Group | 40 mins | 1 |
| | | Weaver Birds | - | Unplugged | Individual or Group | 40 mins | |
| Let's Recognize the Programming Environment | • Adds characters to the screen.<br>• Changes the screen background.<br>• Gives movement to the characters.<br>• Gives movement by changing character costume.<br>• Recognizes the interface and features of the block-based programming tool.<br>• Describes the function of a program in the block-based programming tool. | Leisure Time | - | Plugged | Individual | 160 mins | 2-3 |
| Conditionals | • Analyzes a problem.<br>• Describes the importance of generating authentic solutions for problems.<br>• Discusses the problem-solving process and basic concepts.<br>• Devises an algorithm for solving the problem.<br>• Analyzes different algorithms and chooses the fastest and most accurate solution.<br>• Generalizes the solution to similar problems.<br>• Tests the solution of an algorithm.<br>• Improves and edits a program in the block-based programming tool against given criteria.<br>• Creates programs which include the linear logic structure.<br>• Creates programs which include the condition structure.<br>• Creates programs which include multiple condition structures.<br>• Corrects a program in the block-based programming tool.<br>• Tests and corrects programs involving the linear logic structure.<br>• Tests and corrects programs involving the condition structure.<br>• Tests and corrects programs involving the multiple condition structure. | "Ali and Ayşe are going on holiday" | 1-4 | Unplugged | Individual or Group | 35 mins | 4-5 |
| | | "Colourful Steps" | 4 | Unplugged | Group | 10 mins | |
| | | "Questions and Answers" | 5 | Plugged | Individual | 95 mins | |
| | | "Maze map game" | | | | | |
| | | "Reading and editing codes" | 6 | Plugged | Individual | 20 mins | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | • Divides a problem into sub-problems. | | | | | | |
| Variables | • Analyzes a problem.<br>• Describes the importance of generating authentic solutions for problems.<br>• Discusses the problem-solving process and basic concepts.<br>• Devises an algorithm for solving the problem.<br>• Analyzes different algorithms and chooses the fastest and most accurate solution.<br>• Generalizes the solution to similar problems.<br>• Classifies data by type.<br>• Tests the solution of an algorithm.<br>• Uses the constants and variables to solve the given problem.<br>• Creates programs which include the linear logic structure.<br>• Improves and edits a program in the block-based programming tool against given criteria.<br>• Corrects a program in the block-based programming tool.<br>• Tests and corrects programs involving the linear logic structure.<br>• Divides a problem into sub-problems. | "Buying fruit at greengrocer's" | 1-4 | Unplugged | Individual or Group | 30 mins | 6-7 |
| | | "I choose fruit" | 4 | Unplugged | Group | 10 mins | |
| | | "My fruit basket" | 5 | Plugged | Individual | 95 mins | |
| | | "Reading and editing codes" | 6 | Plugged | Individual | 20 mins | |
| Loops | • Analyzes a problem.<br>• Describes the importance of generating authentic solutions for problems.<br>• Discusses the problem-solving process and basic concepts.<br>• Devises an algorithm for solving the problem.<br>• Analyzes different algorithms and chooses the fastest and most accurate solution.<br>• Generalizes the solution to similar problems.<br>• Tests the solution of an algorithm.<br>• Improves and edits a program in the block-based programming tool against given criteria.<br>• Creates programs which include the linear logic structure.<br>• Creates programs which include the loop structure.<br>• Chooses the most appropriate condition structures to implement an algorithm.<br>• Generates solutions for complex problems by using different programming structures.<br>• Tests and corrects programs involving the loop structure. | "Ali playing a game" | 1-4 | Unplugged | Individual or Group | 30 mins | 8-9 |
| | | "Auto racing" in-class animation | 4 | Unplugged | Group | 10 mins | |
| | | "My fuel amount" | 5 | Plugged | Individual | 95 mins | |
| | | "Auto racing" Application | | | | | |
| | | "Reading and editing codes" Application | 6 | Plugged | Individual | 20 mins | |

- Determines the relation between mathematics and computer sciences.
- Devises an authentic project which involves all programming structures.
- Divides a problem into sub-problems.

All of the activities listed in Table 1 were developed in accordance with the PSS model. Şahin's (2018) thesis provided a reference for developing the lesson plans and activities used in the implementation stage and for testing the suitability of the activities for the students' cognitive levels, determining lesson durations, and coping with overcrowded classrooms. Also, the lesson plans were prepared in accordance with the Directive of the Ministry of National Education on Planned Implementation of Education and Instruction Activities (2003).

## 3. Results

### *3.1. Effects of programming lessons taught with the PSS model on students' satisfaction*

To answer the first research question (What is the effect of applying the PSS model in programming lessons on students' satisfaction with the instruction provided), the researchers analyzed the data collected through students' semi-structured interviews and journals using content analysis, including such topics as "Conditionals," "Variables," and "Loops". Table 2 presents students' reflections on the teaching of "Conditionals".

Table 2. Students' Reflections on Learning about Conditionals through PSS

| **Students' Reflections on Learning about Conditionals through PSS** | **f (n=29)** |
|---|---|
| Lesson was fun. | 19 |
| I'd like lessons in this way. | 5 |
| I find the lesson effective or efficient. I could understand the topic very well. | 16 |
| I find the teaching of the lesson a bit boring. | 2 |

These highly positive responses are supported by the following excerpts:

*Plugged and Unplugged Activities*

> *"I thought it was going to be boring while I was starting. But when I started the class, I started to have fun slowly. I think it was a really fun lesson I enjoyed all the activities; I hope we will go on like this." (K8)*

> *"I think the teaching of the lesson was very nice. Thanks to the activities given to us, the topic was understood very well, and the process was fun." (K11)*

> *"I enjoyed it a lot and had fun. I think it is more efficient for us to be taught in this way. I want it to continue like this." (K18)*

*Plugged Activities*

> *"The plugged activity was far more fun." (K10)*

> *"It gets more fun on the computer." (K4)*

*Unplugged Activities*

> *"I had difficulty understanding the problem in the activity Ali and Ayşe. I think it was difficult, so I was bored."(K15)*

> *"I could understand better the problem and the solution in this way. I realized that I can understand problems more effectively by playing activities and games." (K6)*

> *"It helped me better understand the parts I didn't understand. It was a fun activity."*

*(K20)*

*"I could better understand the activity by seeing, and I believe it will stay longer in my mind in terms of time." (K29)*

*"I think we have visualized the problems thanks to that activity. It also made the lesson more fun." (K31)*

As can be understood from Table 2 and the excerpts above, the students particularly enjoyed learning through the PSS model in connection with "Conditionals." Although some students found some of the relevant activities difficult, they did not make negative comments about the overall instructional process. Only two students said that they were bored by the lesson because they found the problem in unplugged activities difficult to understand. Table 3 presents students' views regarding the teaching of the topic "Variables".

Table 3. Students' Reflections on Learning of Variables through PSS

| **Students' Reflections on Learning of Variables through PSS** | **f (n=30)** |
|---|---|
| Lesson was fun. | 15 |
| I liked that the lesson was taught in this way. | 17 |
| I learnt to devise a strategy. | 3 |
| It was a bit boring. | 4 |

Table 3 demonstrates that students' perceptions of the lesson on variables were highly positive, as exemplified in the following excerpts:

*Plugged and Unplugged Activities*

*"The animation was also nice. It was also easy to transfer this to the Scratch environment." (K31)*

*Plugged Activities*

*"They were very nice and logical activities. I enjoy such activities." (K7)*

*"It was enjoyable and instructive. Maybe it could have been a little harder. Other than that, I liked the faulty code activity the most." (K12)*

*Unplugged Activities*

*"I had fun doing it, the activity was nice and instructive, but it was too easy." (K8)*

*"I couldn't find any negative sides. As for positive, I learned strategy." (K19)*

*"I had a lot of fun; I learned variables better and clearly." (K5)*

*"It was nice. The activity was very easy for me, so I was a little bored." (K3)*

*"I had so much fun. It was a lot of fun to act the fruits and Ali. My thinking ability has improved." (K7)*

*"That was so fun. This game helped me understand variables." (K12)*

As indicated in Table 3 and the excerpts, , "Variables" was enjoyed the most as a result of teaching with the PSS model. However, four students stated that they were bored because they found the problem too easy.

Students' reflections on the topic "Loops" are given in Table 4.

Table 4. Students' Reflections on Learning Loops through PSS

| **Students' Reflections on Learning Loops through PSS** | **f (n=24)** | **%** |
|---|---|---|
| Lesson was fun. | 11 | 45.83% |
| I liked that the lesson was taught in this way. | 13 | 54.16% |
| I could better understand the topic. | 2 | 8.33% |
| It improved my skills of understanding the problem, devising a strategy. | 2 | 8.33% |

Table 4 shows that the students reported substantially positive views regarding the use of the PSS for learning loops. Some of the students' reflections on the process are excerpted below:

*Plugged Activities*

> *"I had more fun on the computer." (K23)*
>
> *"Overall, it was nice. I liked the coding part on Scratch the most. The negative side was that there were no different ideas. Other than that, it was nice." (K8)*
>
> *"It has no negative side. Its positive sides are about our editing the incorrect codes in the activity. In this way, I could understand much better what each operation is for." (K13)*

*Unplugged Activities*

> *"I found it good. It didn't have a negative side. As for the positive side, it helped me devise a strategy and understand better." (K4)*
>
> *"I like it. The positive sides have improved my ability to understand the problem, and the step of creating the algorithm reminded me of the algorithms we have recently created. I don't think there's a negative side." (K6)*
>
> *"The positive sides are about improving our strategy strength. They have no negative sides." (K18)*
>
> *"I found it fun and it made me understand the topic better." (K12)*

As reported in Table 4 and the excerpts, the participants found learning "Loops" using the PSS model to be a pleasant and entertaining experience. Notably, no students reported negative opinions about the course of the lesson.

*3.2. Students' activity type preferences in programming lessons taught with the PSS model along with their justifications*

*Students' responses to "What types of activities do you prefer, plugged only, unplugged only, or both?" are summarized across three topics in Table 5:*

Table 5. Students' Preferences for Activity Type

| Topic | Activity Type | n | f | % |
|---|---|---|---|---|
| Conditionals | Plugged | 30 | 5 | 16.66 |
| | Unplugged | | 5 | 16.66 |
| | Both | | 20 | 66.66 |
| Variables | Plugged | 29 | 7 | 24.13 |
| | Unplugged | | 2 | 6.89 |
| | Both | | 20 | 68.96 |
| Loops | Plugged | 24 | 6 | 25 |
| | Unplugged | | 4 | 16.66 |
| | Both | | 14 | 58.33 |

Table 5 shows that although both types of activities were predominantly preferred, some students opted for only plugged or unplugged activities.

In order to find out the reasons for their preferences, the students were asked *"Why would you prefer a particular type of activity (plugged, unplugged, both) for teaching of the lessons?"* Table 6 summarizes the students' reasons for their preferences along with supporting excerpts from their answers.

Table 6. Students' Reasons for Activity Preferences

| Type of Activity | Reason for Preference | Student's Comments |
|---|---|---|
| Unplugged Activities | (A)<br>• It presents a problem status with more than one path of solution.<br>• The problem has an analogical design.<br>• The solution of the problem offers "Devise a Plan" step.<br>• It types the algorithm for the best plan available.<br>• The plan can be performed with drama activities. | Conditionals – Colourful steps game- *"I could understand the problem and the solution better in this way. I realized that I can understand problems more effectively by activities and playing games." (K6)*<br><br>Conditionals - Colourful steps game- *"It helped me better understand the parts I didn't understand. It was a fun activity." (K20)*<br><br>Conditionals - Colourful steps game- *"I could better understand the activity by seeing, and I believe it will stay longer in my mind in terms of time." (K29)*<br><br>Conditionals- Colourful steps game- *"I think we have visualized the problems thanks to that activity. It also made the lesson more fun." (K31)*<br><br>Variables- Fruit basket- *"I had a lot of fun; I learned variables better and clearly." (K5)*<br><br>Variables- Fruit basket- *"I had so much fun. It was a lot of fun to act the fruits and Ali. What it has brought me is the convenience of doing this application in the Scratch program. My thinking ability has improved." (K7)* |

| | | |
|---|---|---|
| | | Variables- Fruit baskets- *"That was so fun. This game helped me understand variables." (K12)*<br><br>Loops- *"Yes, it was fun. It better taught me the activity." (K4)*<br><br>Loops- *"I found it fun and it made me better understand the topic." (K12)* |
| Plugged Activities | (B)<br>• The visual programming tool Scratch is used.<br>• It corrects the erroneous code structure.<br>• There are interesting activities. | Conditionals - *"The plugged activity was more fun." (K10)*<br><br>Conditionals - *"It becomes more fun on the computer." (K4)*<br><br>Variables- *"They were very nice and logical activities. I enjoy such activities." (K7)*<br><br>Variables - *"It was enjoyable and instructive. Maybe it could have been a little harder. Other than that, I liked the faulty code activity the most." (K12)*<br><br>Loops- *"Overall, it was nice. I liked the coding part on Scratch the most. The negative side was that there were no different ideas. Other than that, it was nice." (K8)*<br><br>Loops - *"I had more fun on the computer." (K23)* |
| Plugged and Unplugged Activities | (A)+(B) | Conditionals- *"The computer activities were as fun as the activities we did in the classroom." (K27)*<br><br>Conditionals- *"Both were fun (with and without a computer), I had a lot of fun." (K28)*<br><br>Variables- *"The activities were fun to me. I had a lot of fun." (K7)*<br><br>Variables- *"They were very amusing and instructive activities." (K11)*<br><br>Loops- *"I think in-class activities are very nice and fun." (K5)*<br><br>Loops- *"The activities were very fun. I had fun." (K6)* |

The results in Table 6 were based on themes obtained from the content analysis. As stated in the right column, some of the students appreciated plugged activities more, while some others favoured unplugged activities in a environment. Some others stated that they liked all the activities in the classroom (including plugged and unplugged) and found them amusing.

## 4. Discussion and Conclusion

This case study was carried out to shed light on students' satisfaction with the plugged and unplugged activities provided in programming lessons using the PSS model and their preferences for activity type along with their reasons. As a result, it was found that some participants preferred plugged activities and others, though not as many, preferred unplugged activities, but most frequent response was that they were positive about both. Furthermore, the study shows that neither plugged nor unplugged activities affect all students in the same way. It can be concluded that our model holds the potential to meet the needs of all students since it encompasses both unplugged and computer activities in the first four and the last three steps, respectively. On the whole, the students' views regarding the learning process with the PSS model using such terms as fun, liking, effective, and

efficient indicate positive effects of the implementation on their interest and motivation.

In the literature, it is reported that students' satisfaction is bound to multiple factors. Sáez-López et al. (2016), in a study on fifth and sixth graders, found out that students were pleased to work with Scratch visual programming tools, enjoyed the activities, found the Scratch programming environment amusing, and felt motivated. Calder (2010) in his study involving sixth graders found that Scratch programming is motivating and interesting for learners. In their research with nine- and ten-year-old students, Taylor, Harlow, and Forret's (2010) also found that plugged activities with the Scratch programming tool were interesting for students. These results are consistent with our finding that students learning programming through the PSS model showed a greater preference for plugged activities more. Using Scratch as a visual and block-based programming tool appeared to increase their interest in and satisfaction with plugged activities and account the findings in the present study. Likewise, previous researchers point out that the use of visual programming tools increases positive reactions such as interest, satisfaction, and amusement (Kalelioğlu, 2015; Kalelioğlu and Gülbahar, 2014; Kelleher et al., 2007; Malan and Leitner, 2007; Pinto and Escudeiro, 2014; Wu et al., 2010).

Previous studies have also established that students enjoy and are motivated by learning through unplugged activities. For instance, Futschek and Moschitz (2011) included entertaining activities at different levels in an unplugged environment for teaching students the basic concepts of algorithm and developing their algorithmic thinking skills prior to switching to a programming environment. In the same vein, Tsalapatas et al. (2012) found that there was an increase in primary students' satisfaction with instruction to develop algorithmic thinking skills when they were offered activities and drama games in an unplugged environment. Moreover, it was noted that including drama activities in ITS lessons beneficial, more amusing, and increased satisfaction with the lessons and beneficial and entertaining (Karaosmanoğlu and Adıgüzel, 2017). Some researchers further defended role playing as a way to lure students to take an active part in the lesson and successfully learn by doing and interacting with peers, which supports collaborative learning (Atalay, 2010).

In addition, the literature suggests that analogies increase the interest, curiosity, and satisfaction of learners (Keller, 1983 as cited by Seyhan, 2015, p.18; Sarıoğlu and Kartal, 2017). Such findings are in congruence with our finding that students learning programming through PSS tend to prefer unplugged activities. It is thought the students in our case favoured unplugged activities mainly because such steps have an analogical basis, drama activities are entertaining, necessary and unnecessary data are separated in the step of understanding the problem, and devising a strategy for a solution to the problem increased students' interest and satisfaction.

Gomes, Falcão and Tedesco (2018) investigated how children can be taught programming concepts with digital games. A primary finding was that students had difficulty understanding the topic of loops and needed unplugged activities to reinforce their understanding of this concept, leading the researchers to recommend including in-class activities besides plugged activities in programming instruction.

Tsarava et al. (2017) conducted a curriculum development study to improve the computational thinking skills of primary school students. In the target curriculum, concepts common to computational thinking, computers, and programming were highlighted. While developing the curriculum, they intended to promote the cognitive operations that underlie computational thinking skills by approaching programming from a cognitive perspective. The curriculum included both plugged and unplugged game-based activities aiming to motivate students.

Leifheit, Jabs, Ninaus, Moeller and Ostermann (2018) also assessed the appropriateness of a game-based approach to teaching programming in primary school to develop students' computational thinking skills such as abstraction, generalization, algorithms, and ability to systematically solve complex problems. In the lessons, algorithm concepts were covered first, and then game-based learning materials were used to introduce the other programming concepts using tangible objects rather than direct coding. In the following lessons, students' learning was reinforced with plugged programming exercises. It was concluded that the game-based, non-digital activity approach had a positive effect on teaching computational thinking and related skills.

Weigend (2014) reports that plugged and unplugged activities together can complement each other and hold students' interest and enthusiasm. Futschek and Moschitz (2011) found that using unplugged exercises at the basic level of teaching programming resulted in students' higher levels of satisfaction for programming on the computer as they experience increasing levels of programming, and even those who had no interest in programming at early stages developed positive attitudes toward programming. Giannakos et al. (2013) pointed out that students' interest in and satisfaction with computer sciences and programming increased as a result of engaging in both unplugged and plugged activities, especially for female students.

Thus the literature supports the effectiveness of a combination of plugged and unplugged activities for enhancing both conceptual understanding of programming and computational thinking skills. It has been fund that unplugged tasks support understanding of a problem, devising and implementing a strategy, creating an algorithm, and dramatizing activities while plugged activities involving the digital media (Scratch, code.org, Lightbot etc.) are motivating.

In this study it was found out that the type of activity had a significant effect on students' satisfaction in the teaching programming as a means of developing cognitive skills. Students' satisfaction increased when they were engaged in programming lessons in their preferred activity type. Thus, combining the two approaches can help students overcome the difficulties they encounter face in programming teaching. Therefore the PSS model, which includes both types of activity, is highly likely to support and appeal to all learners in a programming course.

## 5. Recommendations

For the teaching programming, we recommend that plugged activities should be included in order to

- Determine students' levels of interest in using a computer before starting teaching programming for increase satisfaction with programming, and
- Find out students' views regarding their preferred learning environment.

We recommend unplugged activities that

- include a problem situation that may have more than one solution.
- are formulated analogically.
- Include role-playing or other drama methods.

## References

Ala-Mutka, K. (2004). Problems in learning and teaching programming. *Institute of Soft ware Systems, Tampere University of Technology*. Retrieved June 15, 2018 from https://www.cs.tut.fi/~edge/literature_study.pdf

Arabacıoğlu, T., Bülbül, H. İ., & Filiz, A. (2007, January). A New Approach to Computer Programming Teaching. *In Proceedings of the 9th Academic Informatics Conference*, Dumlupınar University, Kütahya, Turkey.

Aşkar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java programming among engineering students. TOJET: *The Turkish Online Journal of Educational Technology*, 8(1), 26-27.

Asad, K., Tibi, M., & Raiyn, J. (2016). Primary school pupils' attitudes toward learning programming through visual interactive environments. *World journal of education*, 6(5), 20. doi:10.5430/wje.v6n5p20

Atalay, O.(2010). *Effect of the use of drama method on the 5th grade students success in information technologies course*. Unpublished Master Thesis. Gazi University, Education Sciences Institute, Ankara.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. *In Proceedings of the 2012 annual meeting of the American Educational Research Association,* Vancouver, Canada, 1(25).

Brown, Q., Mongan, W., Kusic, D., Garbarine, E., Fromm, E., & Fontecchio, A. (2013). *Computer aided instruction as a vehicle for problem solving: Scratch programming environment in the middle years classroom*. Retrieved September 22, 2017 from http://www.pages.drexel.edu/~dmk25/ASEE_08.pdf

Büyüköztürk, Ş., Çakmak, E. K., Akgün, Ö. E., Karadeniz, Ş., & Demirel, F. (2016). *Bilimsel araştırma yöntemleri*. Ankara: Pegem Akademi.

Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. *ACM SIGCSE Bulletin, 33*(3), 49-52. doi: 10.1145/377435.377467

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

Cohen, L., Manion, L., & Morrison, K. (2005). *Research methods in education* (5th ed.). London: Routledge Falmer.

Coull, N. J., & Duncan, I. M. (2011). Emergent requirements for supporting introductory programming. *Innovation in Teaching and Learning in Information and Computer Sciences*, 10(1), 78-85. doi: 10.11120/ital.2011.10010078

Creswell, J. W. (2013). Steps in conducting a scholarly mixed methods study. DBER Speaker Series. 48.

Çatlak, Ş., Tekdal, M., & Baz, F. Ç. (2015). The Status of Teaching Programming with Scratch: A Document Review Work. *Journal of Instructional Technologies & Teacher Education*, *4*(3), 13-25.

Denner, J., Werner, L., Campe, S., & Ortiz, E. (2014). Pair programming: Under what conditions is it advantageous for middle school students?. *Journal of Research on Technology in Education,* 46(3), 277-296. doi: 10.1080/15391523.2014.888272

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*(1), 240-249. doi: 10.1016/j.compedu.2011.08.006

Erümit A. K., Karal H., Şahin G., Aksoy D.A., Aksoy A., & Benzer A.İ. (2019). A Model Suggested for Programming Teaching: Programming in Seven Steps. *Education & Science*. 44(197), 155-183. doi: 10.15390/EB.2018.7678

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. doi: 10.1016/j.compedu.2012.11.016

Fesakis, G., & Serafeim, K. (2009). Influence of the familiarization with scratch on future teachers' opinions and attitudes about programming and ICT in education. *ACM SIGCSE Bulletin*, 41(3), 258-262. doi: 10.1145/1562877.1562957

Futschek, G., & Moschits, J. (2010). *Developing algorithmic thinking by inventing and playing algorithms*. Retrieved December 9, 2016 from https://publik.tuwien.ac.at/ files/PubDat_187461.pdf

Futschek, G. (2006). Algorithmic thinking: The key for understanding computer science. In: Mittermeir, R. (Ed.), Informatics Education The Bridge between Using and Understanding Computers. Vol. 4226 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, ss. 159 – 168.

Futschek, G., & Moschitz, J. (2011, October). Learning algorithmic thinking with tangible objects eases transition to computer programming. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, 155-164. Springer, Berlin, Heidelberg.

Giannakos, M. N., Jaccheri, L., & Proto, R. (2013, April). Teaching computer science to young children through creativity: Lessons learned from the case of Norway. *In Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research*, Heerlen.

Ginat, D. (2004). On novice loop boundaries and range conceptions. *Computer Science Education*, 14(3), 165-181. doi: 10.1080/0899340042000302709

Gomes, T. C. S., Falcão, T. P., & Tedesco, P. C. D. A. R. (2018). Exploring an approach based on digital games for teaching programming concepts to young children. *International journal of child-computer interaction*, *16*, 77-84. doi: 10.1016/j.ijcci.2017.12.005

Gomes, A., & Mendes, A. J. (2007, September). Learning to program - difficulties and solutions. *In International Conference on Engineering Education*, Coimbra.

Grover, S., Pea, R., & Cooper, S. (2016, March). Factors influencing computer science learning in middle school. *In Proceedings of the 47th ACM technical symposium on computing science education,* New York. doi: 10.1145/2839509.2844564

Hawi, N. (2010). Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. *Computers & Education, 54*(4), 1127-1136. doi: 10.1016/j.compedu.2009.10.020

Helminen, J., & Malmi, L. (2010, October). Jype-a program visualization and programming exercise tool for Python. *In Proceedings of the SOFTVIS '10 Proceedings of the 5th international symposium on Software visualization,* 153-162, Utah, USA. doi: 10.1145/1879211.1879234

Hermans, F., & Aivaloglou, E. (2017, November). To Scratch or not to Scratch?: A controlled experiment comparing plugged first and unplugged first programming lessons. *In Proceedings of WiPSCE '17 the 12th Workshop on Primary and Secondary Computing Education*, 49-56. doi: 10.1145/3137065.3137072

Kafai, Y. B., & Q. Burke. (2014). Connected code: Why children need to learn programming. MIT Press.

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

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, *52*, 200-210. doi: 10.1016/j.chb.2015.05.047

Karaosmanoğlu, G., & Adıgüzel, Ö. (2017). Effects of Creative Drama Method On Students Academic Achievements In Ict Lessons Of Sixth Grades. *Elementary Education Online, 16*(2), 693-712.

Kelleher, C., Pausch, R., & Kiesler, S. (2007, April). Storytelling Alice motivates middle school girls to learn computer programming. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM*, 1455-1464. doi: 10.1145/1240624.1240844

Klopfer, E., & Yoon, S. (2005). Developing games and simulations for today and tomorrow's tech savvy youth. *TechTrends*, 49(3), 33-41. doi: 10.1007/BF02763645

Kobsiripat, W. (2015). Effects of the media to promote the scratch programming capabilities creativity of elementary school students. *Procedia-Social and Behavioral Sciences*, 174, 227- 232. doi: 10.1016/j.sbspro.2015.01.651

Kotsopoulos, D., Floyd, L. M., Khan, S. S., Namukasa, I. K., Somanath, S., Weber, J. L., & Yiu, C. (2017). A Pedagogical Framework for Computational Thinking. *Digital Experiences in Mathematics Education.* 3(2), 154-171. doi: 10.1007/s40751-017-0031-2

Lahtinen, E., Ala-Mutka, K., & Jarvinen, H. (2005) A Study of Difficulties of Novice Programmers. In Acm Sigcse Bulletin, ACM, 37(3), 14-18. doi: 10.1145/1067445.1067453

Lai, C. S., & Lai, M. H. (2012, June). Using computer programming to enhance science learning for 5th graders in Taipei. *International Symposium on computer, consumer and control,* 146-148. Taiwan. IEEE. doi: 10.1109/IS3C.2012.45

Lai, A., & Yang, S. (2011, September). The learning effect of visualized programming learning on 6th graders' problem solving and logical reasoning abilities. *In Electrical and Control Engineering (ICECE), 2011 International Conference on, IEEE*, 6940-6944. doi: 10.1109/ICECENG.2011.6056908

Law, K. M., Lee, V. C., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, 55(1), 218-228. doi: 10.1016/j.compedu.2010.01.007

Leifheit, L., Jabs, J., Ninaus, M., Moeller, K., & Ostermann, K. (2018, October). Programming Unplugged: An Evaluation of Game-Based Methods for Teaching Computational Thinking in Primary School. *In ECGBL 2018 12th European Conference on Game-Based Learning* (p. 344). Academic Conferences and publishing limited.

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM SIGCSE Bulletin*, 39(1), 223-227. doi: 10.1145/1227310.1227388

Mannila, L., Peltomäki, M., & Salakoski, T. (2006). What about a simple language? Analyzing the difficulties in learning to program. *Computer Science Education*, *16*(3), 211-227. doi: 10.1080/08993400600912384

McMillan, J. H. (2000). *Educational research: Fundamentals for the consumer* (3th ed.). New York: Longman.

Merriam, S. B., & Tisdell, E. J. (2015). *Qualitative research: A guide to design and implementation*. San Francisco: John Wiley & Sons.

Milková, E., & Hulkova, A. (2013). Algorithmic and logical thinking development: base of programming skills. *WSEAS Transactions on Computers*, *12*(2), 41-51.

Naharro-Berrocal, F., Pareja-Flores, C., Urquiza-Fuentes, J., & Velazquez-Iturbide, J. A. (2002). Approaches to comprehension-preserving graphical reduction of program visualizations. In Proceedings of the 2002 ACM symposium on Applied computing, ACM, 771-777. doi: 10.1145/508791.508941

Özmen, B., & Altun, A. (2014). Undergraduate students' experiences in programming: difficulties and obstacles. *Turkish Online Journal of Qualitative Inquiry*, 5(3), 1-27. doi: 10.17569/tojqi.20328

Pinto, A., & Escudeiro, P. (2014, June). The use of scratch for the development of 21st century learning skills in ICT. Information Systems and Technologies, *9th Iberian Conference*, (pp. 1-4). IEEE. doi: 10.1109/CISTI.2014.6877061

Qian, Y., & Lehman, J. D. (2016). Correlates of success in introductory programming: A study with middle school students. *Journal of Education and Learning*, *5*(2), 73. doi: :10.5539/jel.v5n2p73

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137-172. doi: 10.1076/csed.13.2.137.14200

Sarıoğlu, T., & Kartal, G. (2017). Drama as Method—An Alternative in Teaching Information and Communication Technologies? *Elementary Education Online, 16*(1), 366-376.

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education*, *97*, 129-141. doi: 10.1016/j.compedu.2016.03.003

Schwartz J., Stagner J., & Morrison, W. (2006). Kid's programming language (KPL). *In ACM SIGGRAPH 2006 Educators program, ACM.* doi: 10.1145/1179295.1179348

Seppälä, O., Malmi, L., & Korhonen, A. (2006). Observations on student misconceptions—A case study of the Build–Heap Algorithm. *Computer Science Education*, 16(3), 241-255. doi: 10.1080/08993400600913523

Seyhan, H. G. (2015). Okul öncesi fen eğitiminde analoji kullanımının önemi ve analoji örnekleri. *Cumhuriyet International Journal of Education, 4*(2), 15-28.

Shih, I.-J. (2014). *The effect of scratch programming on the seventh graders' mathematics abilities and problem solving attitudes.* Unpublished master's thesis. Taipei University, Taiwan.

Silverman, D. (2013). *Doing qualitative research: A practical handbook*. London: SAGE Publications.

Stolee, K. T., & Fristoe, T. (2011, March). Expressing computer science concepts through Kodu game lab. *In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 99-104. Dallas, USA. doi: 10.1145/1953163.1953197

Şahin, G. (2018). *A methodology for teaching programming at middle school level.* Unpublished Master Thesis. Karadeniz Technical University, Education Sciences Institute, Trabzon.

Taylor, M., Harlow, A., & Forret, M. (2010). Using a computer programming environment and an interactive whiteboard to investigate Some Mathematical Thinking. Procedia Social and Behavioral Sciences, 8 (2010), 561-570. doi: 10.1016/j.sbspro.2010.12.078

Tsalapatas, H., Heidmann, O., Alimisi, R., & Houstis, E. (2012). Game-based programming towards developing algorithmic thinking skills in primary education. *Scientific Bulletin of the" Petru Maior" University of Targu Mures*, *9*(1), 56-63.

Tsarava, K., Moeller, K., Pinkwart, N., Butz, M., Trautwein, U., & Ninaus, M. (2017, October). Training computational thinking: Game-based unplugged and plugged-in activities in primary school. In *European Conference on Games Based Learning*, 687-695. Nijmegen, Netherlands.

Wagner, T. (2008). *The global achievement gap: Why even our best schools don't teach the new survival skills our children need-and what we can do about it.* New York: Basic Books.

Weigend, M. (2014). The digital woodlouse--scaffolding in science-related scratch projects. *Informatics in Education*, 13(2), 293-305. doi: 10.15388/infedu.2014.09

Wilson, A., & Moffat, D.C. (2010, September). Evaluating Scratch to introduce younger school children to programming. *In Proceedings of the 22nd Annual Psychology of Programming Interest Group,* Leganés, Spain.

Wu, W., Chang, C., & He, Y. (2010). Using Scratch as game-based learning tool to reduce learning anxiety in programming course. *In Proceedings of Global Learn Asia Pacific*, 1845- 1852.

Yıldırım, A., & Şimşek, H. (2005). *Sosyal bilimlerde nitel araştırma yöntemleri.* Ankara: Seçkin Yayıncılık.

Zanetti, H., Borges, M., & Ricarte, I. (2016, November). Pensamento Computacional no Ensino de Programação: Uma Revisão Sistemática da Literatura Brasileira. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE),* 27(1), 21. doi: 10.5753/cbie.sbie.2016.21

**Appendix A**

**Semi-Structured Interview Questions for Informatics Technologies and Software Teachers**

INTERVIEW QUESTIONS REGARDING THE LESSON PLAN TEACHING PROGRAMMING TO MIDDLE SCHOOL 6TH GRADE STUDENTS

Interviewee: …… Interviewer:…….

Date and Time: ……/……2017 at ……..:……

Duration: ……

1. How suitable do you think the activity in the lesson plan for teaching programming to middle school 6th graders is for the students' levels?
2. How suitable do you think the computer activities in the lesson plan for teaching programming to middle school 6th graders are for the students' levels?
3. Do you find the duration allocated for the activities in the lesson plan for teaching programming to middle school 6th graders sufficient? Why or why not?
4. Do you find the duration allocated for the computer activities in the lesson plan for teaching programming to middle school 6th graders sufficient? Why or why not?
5. How applicable do you think the activities in the lesson plan for teaching programming to middle school 6th graders are in overcrowded classrooms?
6. How applicable do you think the computer activities in the lesson plan for teaching programming to middle school 6th graders are in overcrowded classrooms?

**Appendix B**

**Students' Diaries on Problem Solving Concepts and Approaches**

| |
|---|
| 1. On the whole, how did you find the teaching of the lesson? (Please specify the good and bad sides, and what you liked and disliked.) |
| 2. What are the good sides of the activities in the lesson? |
| 3. What are the bad sides of the activities in the lesson? |
| 4. What do you think an algorithm resembles? |
| 5. Remembering the activities performed in the lesson, what do you think an algorithm is? |
| 6. Please discuss the activities about algorithm which were performed in the lesson. |
| 7. What do you think a problem resembles? |
| 8. Remembering the activities performed in the lesson, what do you think a problem is? |
| 9. Please discuss the activities about the problem status which were performed in the lesson. |

**Appendix C**

**Activity: "Helping the Shepherd"**

Scenario: There was a shepherd. Next to the shepherd was a wolf, a grass and a sheep. They'il cross the river with a raft. If the wolf passes the sheep eats grass, the grass passes, the wolf eats the sheep, the sheep wolves, the sheep crosses without eating sheep.

**Appendix D**

**Activitity: "I am finding the cities"**

Scenario: Friends, we see the map engineer Ayşe on the picture. Her new job is to map the cities in the region where a circular lake is located. Can you help her with this? The distances between these cities are given in the table. Match cities A, B, C, D and E in the table with the numbers 1, 2, 3, 4 and 5 in the photo.

**Appendix E**

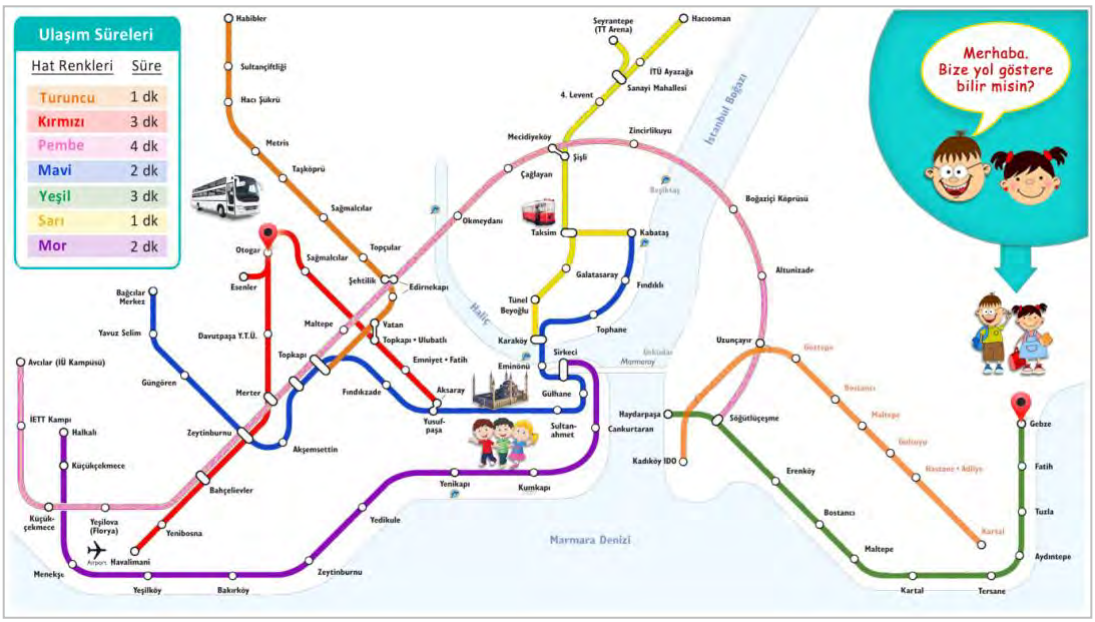
**Guidelines to Examination Scratch Projects for Students**

| Category | Reviewed application name | Feature | | | | |
|---|---|---|---|---|---|---|
| | | Movement | Sound | Changing scene | Talking balloon | Move with keyboard |
| Games | | | | | | |
| Animations | | | | | | |
| Stories | | | | | | |
| Music and dance | | | | | | |
| Art | | | | | | |
| Your application idea | | | | | | |

**Appendix F**

**Activitity: "Ali and Ayşe are going on holiday"**

Scenario: Ali and Ayşe live in Gebze. They'il go to their grandmother with bus. But before they go, they want to say goodbye to their friends. So, they will meet their friends in Sultanahmet Square. But they don't know which way to go. Can you show Ali and Ayşe the way to Sultanahmet and then to the bus terminal as soon as possible?



**Appendix G**

**"Ali and Ayşe are going on holiday" activity Problem Acknowledgement Table (PAT) form**

| # | Questions | Answers |
|---|---|---|
| **1.** | Where do Ali and Ayşe live? | Gebze |
| **2.** | Who will Ali and Ayşe go to? | Grandmother |
| **3.** | Where will Ali and Ayşe meet with their friends and where will they go after they meet? | They will meet at Sultanahmet Square. They'il go to the bus station after saying goodbye to their friends. |
| **4.** | How do you describe the transfer points on the map? | The intersection of two different colors the path. |
| **5.** | What do transportation times mean at the event? | The elapsed time between two stops in each line color. |
| **6.** | How is the time elapsed from one stop to another stop in the activity calculated? | With given transportation times. |

**Appendix H**

**"Ali and Ayşe are going on holiday" activity Strategy Devise Form (SDF)**

| # | Boarding Stop | Landing Stop | Count of Stops | Line Color | Time |
|---|---|---|---|---|---|
| **1.** | Gebze | Söğütlü Çeşme | 9 | Green | 27 min |
| **2.** | Söğütlü Çeşme | Mecidiyeköy | 5 | Pink | 20 min |
| **3.** | Şişli | Karaköy | 4 | Yellow | 4 min |
| **4.** | Karaköy | Sultanahmet | 4 | Blue | 8 min |
| **5.** | Sultanahmet | Yusuf-paşa | 1 | Blue | 2 min |
| **6.** | Aksaray | Otogar | 4 | Red | 12 min |
| **7.** | | | | | |
| **8.** | | | | | |
| **9.** | | | | | |
| **10.** | | | | | |
| | | | | **Total Time:** | 73 min |
| | | | **Total Number of Transfers:** | | 4 |