Received: 6 October 2021 | Revised: 14 November 2022 | Accepted: 4 December 2022

DOI: 10.1111/jcal.12771

ARTICLE

Journal of Computer Assisted Learning WILEY

The effects of block-based visual and text-based programming training on students' achievement, logical thinking skills, and motivation

Şenol Saygıner[1] | Hakan Tüzün[2]

[1]Faculty of Education, Department of Computer Education and Instructional Technology, Hatay Mustafa Kemal University, Hatay, Turkey

[2]Faculty of Education, Department of Computer Education and Instructional Technology, Hacettepe University, Ankara, Turkey

Correspondence
Şenol Saygıner, Faculty of Education, Department of Computer Education and Instructional Technology, Hatay Mustafa Kemal University, Antakya/Hatay 31060, Turkey.
Email: senolsayginer@gmail.com

Abstract

Background: Studies on the effectiveness of block-based environments continue to produce inconsistent results. A strong reason for this is that most studies compare environments that are not equivalent to each other or to the level of learners. Moreover, studies that present evidence of the effectiveness of block-based environments by comparing equivalent environments are limited.

Objectives: This study aims to scrutinize the effects of programming training to be held in equivalent environments (block-based and text-based) with university students who do not have prior programming knowledge and experience on achievement, logical thinking, and motivation.

Methods: The study was conducted by using an experimental pretest-posttest control group design. The study was conducted with 60 students, the total consisting of 30 students in the experimental group and 30 students in the control group. In the experimental group, block-based visual programming training with Scratch was conducted and the control group received text-based programming training with Small Basic. The training was maintained for 10 weeks, for 4 h a week in each group. The programming achievement test, the logical thinking skills test, and the motivation scale were used to collect the data.

Results and Conclusions: The results showed that the use of a block-based environment in programming training contributed positively to the development of students' logical thinking skills, and motivation for learning programming. In contrast, there was evidence that this training did not make a difference on programming success.

Implications: The findings of the study provided evidence of the effectiveness of block-based training in comparisons made in equivalent environments. Focusing research on this issue may contribute to the improvement of the current understanding.

KEYWORDS
Programming, programming environments, achievement, logical thinking skills, motivation

1 | INTRODUCTION

Developments in today's information and communication technologies force students to change the competencies that they should have. These students, also called "digital natives" or "21st-century individuals" in this period, are expected to come up with analytical solutions to problems and to think systematically and alternatively (ISTE, 2016). Most researchers explained that programming could be a

way to gain individuals those skills (Clements & Gullo, 1984; Deng et al., 2020; Mains, 1997; Malik, 2018; Monroy-Hernandez & Resnick, 2008; Nouri et al., 2020; Swain, 2013; Zhang & Nouri, 2019).

Learning programming, however, is not easy for novices (Espinal et al., 2022; Jesennia et al., 2020; Quille & Bergin, 2019; Topallı & Çağıltay, 2018; Yusoff et al., 2020). The programming process is complex, requiring attention to a variety of elements (such as syntax, algorithm design, error detection, etc.) simultaneously (Saygıner & Tüzün, 2018). Syntax is a critical element among those. Syntax represents the more mechanical components of programming (for example, upper/lowercase letters, opening/closing parentheses, quotation marks, spaces, looping characters, etc.). Denny et al. (2011, 2012) stated that even experts in programming have difficulties in syntax and this situation negatively affects their ability to create algorithms. Another study pointed out that students' efforts to debug syntax may cause cognitive overload (Lye & Koh, 2014). Cognitive overload, then, may have a negative effect on students' learning performance (Paas et al., 2004). Additionally, students who learning environments with complex syntax rules (Java, C, Python, etc.) will likely be disappointed in the learning process and their enthusiasm will likely decline over time (Topallı & Çağıltay, 2018).

Block-based educational environments such as Scratch, Alice, and MIT App Inventor have been developed for students to learn programming more easily (Lye & Koh, 2014). Coding occurs with a drag-and-drop action in those environments. A code block is completed by bunching semantically appropriate blocks together. The related literature asserted that those environments remove the constraints of the mechanical and complex nature of programming and facilitate students' focus on logic and structures (Armoni et al., 2015; Noone & Mooney, 2018). However, experimental studies yielded inconsistent results. Some studies argue that more effective results are achieved in block-based environments when compared to text-based ones in terms of programming learning, development of cognitive skills, and continuity of motivation (Costa & Miranda, 2019; Hongwarittorn & Krairit, 2010; Lim & Kim, 2019; Pratiwi et al., 2018; Weintrop, 2019). Others evidenced that text-based programming environments were professional programming languages (Espinal et al., 2022), block-based environments did not make a difference in the performance of programming learning (Mihci & Ozdener Donmez, 2017) and students gained poor skills in block-based environments (Moors et al., 2018).

Studies on the effectiveness of block-based environments present inconsistent results. Hu et al. (2021) stated in their meta-analysis study that such inconsistent results may be due to four factors: (1) the structure of the selected programming environment, (2) learners' level, (3) the design of the experimental research, and (4) the location of the school (difference by country). The first two factors can be argued to be more critical in programming learning. There is a high possibility that the literature contains many experimental studies comparing environments that do not correspond to learners' levels or are not suitable/equivalent to each other. That is, there are studies comparing Scratch or Code.org with its simple and fun interface in one group, while the other compares Java or C, which are high-level programming environments. Comparing non-equivalent environments is likely to affect research results. The critical issue here is deciding on whether the chosen programming environments, especially for novices, are suitable for both each other and the level of the learner. Nonetheless, experimental studies investigating the effectiveness of teaching programming through attending to that critical issue are limited in number (Hu et al., 2021; Kandin & Sendurur, 2022; Tijani et al., 2020; Weintrop, 2019; Xu et al., 2019). To this end, this article aims to contribute to this limitation in the literature. In other words, this study aims to scrutinize the effects of programming training to be held in equivalent environments (educational programming environments) with university students who do not have prior programming knowledge and experience on achievement, logical thinking, and motivation. Accordingly, the three research questions (RQs) were formulated as follows:

RQ1: What is the effect of block-based and text-based programming training on students' achievement?

RQ2: What is the effect of block-based and text-based programming training on students' logical thinking skills?

RQ3: What is the effect of block-based and text-based programming training on students' motivation levels?

The article presents the results of a quasi-experimental study designed to improve our understanding of these research questions. In this study, in which block-based programming with Scratch was taught in the experimental group and text-based programming with Small Basic in the control group for 10 weeks, all processes were carried out by the same researcher and under the same conditions. In the following section, information and research on programming learning, logical thinking, and motivation variables are presented. It also includes evaluations of block-based programming. Then, information about the method of this study is submitted. Subsequent to the method, the data and results obtained from the research are explained. The article ends with a discussion of the findings on the effectiveness of block-based programming teaching, possible results, limitations, and future studies.

2 | THEORETICAL FRAMEWORK AND RELEVANT LITERATURE

2.1 | Learning programming

Programming is the process of compiling and running command sequences required for any solution to a problem after transforming them into commands that can be understood by the computer (Nouri et al., 2020). In other words, programming is a process that creates the relation between hardware and software. Deciding on how the hardware will respond and guide the behaviour is not easy. Especially for novices, the complex and abstract structure of programming environments, the obligation to comply with syntax rules, and limitations

such as interaction and flexibility make the process even more challenging for students (Quille & Bergin, 2019).

Certain initiatives are taken to make programming learning easy and understandable. The milestone of these initiatives can be the construction of the Logo programming language. Logo has been developed with practical commands so that students can develop applications without having to learn the complex code structures of traditional programming languages (Papert, 2013). This language, which has a text-based structure, provides features such as an interface appropriate for the development level of students, writing code with gradual steps, and providing concrete and visual feedback (Mladenovic et al., 2018). Papert, the developer of this programming language, based the Logo programming language on two theories: (1) students construct knowledge instead of taking it ready-made, and (2) the activity of learning is more meaningful when it is based on authentic experiences (Harel & Papert, 1991). On this basis, Papert's ideas can be stated to be similar to Piaget's and the Logo programming language he developed includes traces of constructivist philosophy.

Today, the Logo language, which Papert developed based on the theory he advocated, has begun to revive with the development of educational programming environments. Educational programming is an approach that advocates for students of all ages to gain programming knowledge and skills rather than developing computer applications that will find solutions to complex problems encountered in real life (Kandemir, 2018). This approach can be basically evaluated under two headings. Text-based ones (Microsoft Small Basic, Python, Greenfoot, Codemonkey, Pencil Code, etc.) are such environments that have all the principles of high-level programming languages (Java, C, C++, etc.) but offer it in a simpler way. That is, these environments also present such elements as the act of writing code, syntax rules, and debugging. Yet, students are exposed to those at a lower level of difficulty.

Another educational approach that aims to facilitate programming learning and has traces of Papert's educational philosophy is block-based programming environments. These environments encourage students to develop something and turn their ideas or discoveries into products by turning coding into a drag-and-drop activity (Resnick, 2012). The focal point of learning in these environments is semantics rather than syntax (Mladenovic et al., 2018). To put it differently, the codes which consist of blocks of different shapes and colours reduce students' efforts to remember the code, put their ideas into code, or memorize the syntax (Weintrop & Wilensky, 2019).

There are many block-based programming environments (Scratch, Alice, App Inventor, etc.) that correspond to the aforementioned characteristics. However, Scratch stands out more than its peers (Baz, 2018). Developed with the idea of Papert and Solomon (1971) that "students should learn to program their own animations, games, and simulations", Scratch offers students a rich space in which they can express themselves (Resnick, 2012). The prominent characteristics of Scratch that distinguish it from text-based environments are the ability to put blocks together by drag-and-drop method, the absence of syntax rules, its simplified language, its interactive structure, its open-source code structure, having blocks of different colours and shapes, and providing an opportunity to participating in an online community and sharing and to co-creation.

Another distinguishing characteristic of Scratch is that this environment can work in a "tinkering learning" system. Tinkering learning, which is also called "bricolage thinking", prioritizes students to develop something through "experience", "experiment", and "discovery" (Berland et al., 2013). Individual and collective productions are important in this way of thinking (Martinez & Stager, 2013). Therefore, it is a synthesis of both the cultures of do it yourself (Do-It-Yourself-DIY) and do it with others (Do-It-With-Others-DIWO). Scratch can turn the theories advocated by tinkering learning (experience, experiment, discovery, producing something individually or together) into practice. To illustrate, every new code block added to the program can be tested instantly in Scratch. Being able to see the result of each manipulation immediately encourages students to build relationships between their actions and results. This cause-effect connection established in tinkering activities promotes the development of students' logical reasoning skills (Cınar et al., 2019). Programs developed with Scratch can be transferred to online environments. Thus, program developers can notice each other's projects or contribute to them (Hagge, 2017). In addition, visual clues (code blocks of different colours and shapes) in this environment reduce the problems experienced during the process of coding (such as syntax, incorrect coding, and punctuation) (Durak, 2020), which leads students to attend more to discovering something while coding.

The related literature includes a large number of studies utilizing the Scratch environment. Such a study was conducted by Korkmaz (2016) with university students. The treatment group received Scratch and block-based training while the control group was taught text-based programming with C++, one of the high-level languages in this study that lasted for 6 weeks. The change in the achievement scores of both groups was scrutinized and the results indicated a higher increase in the achievement scores of the treatment group. Another study with university students conducted by Cardenas-Cobo et al. (2021) also presented the results of an experimental study, in which the treatment group that received programming training with Scratch showed higher achievement compared to the group that received training in the Java environment. These two studies compared educational programming environments and high-level programming environments. The complexity of high-level programming environments may have led to less effort by students in this group. This situation may have caused the difference in achievement in favour of the Scratch group. The results of Akbay's (2019) study also presented remarkable results in that regard. Unlike the previous two studies, Akbay's study did not compare Scratch with high-level programming environments (C++, Java, etc.), but rather compared it with an educational text-based environment (Phyton). This study, which was conducted with university students without programming experience revealed that the achievement of the group that received training in the Python environment (text-based) was higher than the group that studied with the Scratch environment. The results make asking the

question of whether the gap between block- and text-based environments are closing more meaningful. However, the predictions on this issue are mostly at the level of theory since empirical studies comparing educational programming environments in terms of effectiveness have certain limitations. The research to be conducted aims to improve our current understanding of recognizing the domain of educational programming environments.

2.2 | Logical thinking skills

Logical thinking is the sequence of mental operations that a person uses when encountered with a problem (Karplus, 1977). Mental processes such as recognizing abstract structures, using ways of reasoning (induction, deduction), distinguishing similarities and differences between situations, and making logical decisions by making various comparisons and inferences form the basis of logical thinking (Lazear, 2000). Similar mental processes also take place in programming. To illustrate, students are expected to first prepare a plan (algorithm) so as to solve the problem they encounter in programming training. It is of utmost importance to design the solution step by step because a small mistake in the plan will change the overall operation of the program. Wherefore, students have to actively use cognitive skills such as establishing cause-effect relationships, making comparisons, multidimensional thinking, and decision making in learning programming.

The fact that logical thinking and programming share similar processes may at first glance suggest that there is a positive relationship between them. Yet, most studies conducted in previous years indicate otherwise. As such, the study conducted by Mains (1997) examined whether there was a change in the logical thinking skills of university students while learning programming. The text-based programming was taught with QBasic for 14 weeks in this study. The results showed no significant relationship between programming learning and the development of logical thinking skills. Another study by Seidman (1981), in which students were trained to use the Logo programming language also revealed no significant change in the students' logical thinking scores. Similar results were observed in the studies of Clements and Gullo (1984). It is not surprising to reach such results in these periods when educational programming approaches are still developing and mostly high-level text-based environments are used. Besides, studies yielded a lot of evidence that the abstract and complex nature of text-based environments reduces students' desire and effort to learn (Bayman & Mayer, 1983; Byrne & Lyons, 2001; Kinnunen & Malmi, 2008).

The use of educational programming environments (block-based and text-based) has become widespread at the beginning of the 21st century so as to make learning programming more comprehensible. Scratch is one of these environments and can be effective in encouraging students to experience this process with its aspects that prioritize active learning, interaction, and discovery. Students' more programming experience increases their likelihood of making more logical inquiries. The existing literature evidenced the positive effect of this experience on cognitive skills (Durak, 2020; Papadakis & Kalogiannakis, 2019; Weintrop, 2019). In a study by Pratiwi et al. (2018), a group of novice university students received programming training with Scratch, and then, the change in their logical thinking skills was investigated. The results showed a significant increase in their logical thinking skills. This study emphasized the possible reason for this increase, which could be due to the fact that it is easier to put the algorithm into code in the Scratch environment. In a similar vein, Fidai et al. (2020) conducted a meta-analysis study, in which a total of 12 experimental studies were examined. The results illustrate that educational programming with Scratch fostered the development of cognitive skills. This study also highlights the need for more experimental research on scrutinizing the effectiveness of programming with Scratch. Regarding the features in Scratch that are constantly updated to make programming learning easier (Scratch 1.0, Scratch 2.0, Scratch 3.0, etc.), the suggestion highlighted in this research is worth to be taken into account. Identifying which of the features in Scratch makes students more cognitively active can prevent critical points from being over looked when designing the learning process.

2.3 | Motivation to learn programming

Some students are quite eager to find a solution to a problem they face while others are quite reluctant to deal with the problem. The most important factor causing such a variation is motivation (Schunk et al., 2014). Motivation is a driving force that stimulates individuals' actions, directs their behaviour, and determines and maintains the level of behaviour (Deci et al., 2001). Motivation is shaped internally by individuals and externally by environmental factors (Ryan & Deci, 2000). The internal factors are usually individuals' attitudes, expectations, goals, and emotions. The external factors are the desire to learn through reward, competition, avoidance of punishment, and social pressure, which emerges under the influence of the environment (Dede & Argün, 2004; Ryan & Deci, 2000). These two factors are considered to be activators for learning and academic achievement (Jenkins, 2001). Students with high motivation are stated to be more successful in terms of participation in the task, being able to maintain working longer, and learning performance (Pintrich, 2003).

Motivation is a process that includes physical and mental activities rather than being a product (Schunk et al., 2014). Psychical activities include effort, perseverance, and other obvious actions while mental activities include planning, reasoning and decision making, and so forth. Programming is one of the topics that require students to be motivated both physically and mentally as it is inevitable that students will be exposed to working in front of the screen for a long time or make intense mental inquiries in learning programming. Students' motivation should be constantly fostered to increase their performance in this course, where practice-based activities are frequently included (Jenkins, 2001).

A growing body of literature focuses on techniques to support students in learning programming, in that visual programming environments (Ruf et al., 2014), storytelling (Kelleher et al., 2007), and even

singing (Siegel, 1999) are among those techniques. Visual programming environments make abstract programming concepts such as "variable, loop, array" more concrete by supporting them with multimedia-based activities. Studies present promising results in terms of learning programming are more motivating with those techniques. As such, studying with university students, Quahbi et al. (2015) observed the change in students' motivation scores in their study, in which block-based (treatment) with Scratch and text-based (control) programming with Pascal were taught to students who did not have programming experience before. This study reported a greater increase in the motivation of the students that received training about Scratch when compared to those who received text-based training. The rate of those who find the Pascal environment boring and monotonous in the study is 79% while this rate is 15% for the Scratch environment. Students who were training in the Scratch environment stated that they could see each other's work and that creating games and stories in this environment made them creative and autonomous. This study explained the possible reasons for the differences in their motivation with these statements. Another study conducted by Erol and Kurt (2017) with university students gave training about using Scratch to the treatment group and the control group was taught programming with flow diagrams for 7 weeks. In the next 7-week period, the change in motivation and achievement scores was scrutinized by teaching both groups with a text-based program with C#. The results informed that the motivation scores of the participants in the control group decreased at the end of the whole application, and the scores increased in the treatment group. The results also revealed a significant difference in the achievement scores in favour of the treatment group. This study explained the higher achievement in the treatment group with the motivation factor; that is, the students in this group are more motivated to programming. To conclude, this study put forward a theory that learning programming with Scratch leads to increased motivation and increased motivation leads to higher achievement. This study also proposed further research for experimental testing of this theory.

3 | METHOD

3.1 | Research design

This is a quantitative study that used the control group research model, a quasi-experimental design (Fraenkel et al., 2012). In this method, the researcher carries out comparable processes followed by the examination of the effects of those processes. In this context, two classes were determined randomly, one of which was the experimental group and the other one the control group. In the experimental group, block-based visual programming training with Scratch was conducted and the control group received text-based programming training with Small Basic. These two environments are educational programming environments, and it can be stated that they are equivalent to each other in terms of level. The data collection tools which were the programming achievement test, the logical thinking skills test, and the motivation scale were applied two times, one before the training and the other after the training. Information about the design pattern of the research is presented in Table 1.

3.2 | Study group

The study group included sophomore students studying in the Department of Mathematics and Science Education in a state university in Turkey during the spring semester of the 2015–2016 academic year. The study was conducted with 60 students, the total consisting of 30 students (eight males and 22 females) in the experimental group and 30 students (10 males and 20 females) in the control group. Descriptive information regarding the study group is presented in Table 2.

As seen in Table 2, students in the study group have similarities both in gender and in the school type from which they graduated. The distribution regarding their high school or former education or personal developmental education in algorithms, coding, or programming indicates that none of the students had previous education related to programming.

3.3 | Data collection tools

In this study, the Programming Achievement Test, Logical Thinking Skills Test, and Motivation Scale were used as the data collection tools.

3.3.1 | Programming achievement test

The programming achievement test (PAT) used in the study was developed by the researchers. A repository of 30 questions in classical and multiple-choice format was created. Questions were submitted to four academicians who were engaged in programming education in the Department of Computer Sciences at two universities and to one academician for language usage and feedback. At this stage, 10 questions were eliminated for such reasons as "having similar questions, questions inappropriate for the audience, and being unable to create choices" and the test was reduced to 20 questions (four classical, 16 multiple-choice). The constructed test was applied to 174 students, who were the ones studying in the Department of Mathematics and Science Education where the study was conducted, and the ones who had received previous programming education. The students were grouped according to the scores they had on the test, and statistical analysis was conducted by choosing 27% of the subgroup and 27% of the super group to run an item analysis. After the analysis, the seventh (0.11) and 14th (−0.02) questions were eliminated, and the number of questions was reduced to 18.

After the unsuitable items were removed from the achievement test applied to 174 students, the reliability coefficient of the final test was calculated, and the KR-20 value was found to be 0.80. This result

TABLE 1 Research design

Groups	Pretest	Process	Posttest
Experimental	PAT_{pre}, $LTST_{pre}$, Int MOT_{pre}, Ext MOT_{pre}	BBVP-S	PAT_{post}, $LTST_{post}$, Int MOT_{post}, Ext MOT_{post}
Control	PAT_{pre}, $LTST_{pre}$, Int MOT_{pre}, Ext MOT_{pre}	TBP-SB	PAT_{post}, $LTST_{post}$, Int MOT_{post}, Ext MOT_{post}

Abbreviations: BBVP-S, block-based visual programming – Scratch; Ext MOT_{post}, extrinsic motivation – posttest; Ext MOT_{pre}, extrinsic motivation – pretest; Int MOT_{post}, intrinsic motivation – posttest; Int MOT_{pre}, intrinsic motivation – pretest; $LTST_{post}$, logical thinking skills test – posttest; $LTST_{pre}$, logical thinking skills test – pretest; PAT_{post}, programming achievement test – posttest; PAT_{pre}, programming achievement test – pretest; TBP-SB, text-based programming – small basic.

TABLE 2 Descriptive information regarding the study group

		Experimental group		Control group	
	Specification	*f*	%	*f*	%
Gender	Male	8	27	10	33
	Female	22	73	20	67
The school type graduated from	Common High School	19	64	18	60
	Anatolian High School	10	33	12	40
	Anatolian Teacher High School	1	3	–	–
Any former education related to coding, algorithms, or programming?	Yes	0	0	0	0
	No	30	100	30	100

demonstrates the high reliability of the test. The mean difficulty index of the test was calculated as 0.45. This value shows that the test has an average difficulty. The mean discrimination index of the test was calculated as 0.55. Since it is close to the preferred discrimination index (0.50), the test is claimed to be quite discriminant. Because the *z* statistic, which was obtained by dividing the coefficient of skewness by the standard error, was smaller than 1.96 for $\alpha = 0.05$, it was not considered that the distribution showed an extreme deviation from normal (Büyüköztürk, 2007). In accordance with the data being lower than 1.96 for $(0.14/0.25 = 0.56)$, it was shown that an achievement test with normal deviation was developed.

3.3.2 | Logical thinking skills test

The logical thinking skills test (LTST) was developed by Tobin and Capie (1981) and consists of 10 two-stage questions that measure five logical operations named controlling variables, proportional thinking, contingent thinking, relational thinking, and integrative thinking. The test questions require primarily selecting an answer from a set of options, and then choosing the reason for the answer from the given options. To accept a response as correct, both of these stages need to be marked and both need to be answered correctly. The reliability coefficient of the test was reported as 0.85 by Tobin and Capie (1981). Turkish translation and customization of the test was conducted by Geban et al. (1992), and the Cronbach's Alpha reliability coefficient was calculated as 0.77. The maximum score of the test is 10. A student with a score in the range of 0–3 is rated as low, a student with a score in the range of 4–6 is rated as medium, and a student with a score of 7–10 is considered to have a high level of logical thinking (Oliva, 2003).

3.3.3 | Motivation scale

The scale used to determine motivation toward learning programming was developed by Pintrich et al. (1991) with the title "Motivated Strategies for Learning Questionnaire" (MSLQ). The adaptation of the motivation scale (MOT) into Turkish was conducted by Büyüköztürk et al. (2004). The validity and reliability of the Turkish version of the scale were ensured by using the data obtained from 852 university students who were sophomores, juniors, and seniors.

The scale consists of six subfactors and 31 items in total. For this study, only two subfactors of the scale were used, one of which is intrinsic motivation and the other extrinsic motivation. Studies draw attention to the fact that it will be difficult for students to learn programming unless they are internally or externally motivated (Jenkins, 2002; Ryan & Deci, 2000; Souza & Bittencourt, 2019) since intrinsic and extrinsic motivations are the most important factors explaining programming learning (Jenkins, 2001; Jenkins, 2002). Thus, this article is built on the intrinsic and extrinsic dimensions of the motivation variable.

The intrinsic motivation portion of the scale covers inner tendencies such as a student's interest in programming and enjoying programming, while the extrinsic motivation portion covers outer tendencies such as competition, reward, and the like. The scale includes four items to measure intrinsic motivation and four items to measure extrinsic motivation. A minimum of 4 and a maximum of 28 score can be obtained for each intrinsic and extrinsic motivation factor. A high score that is gained from any of the two factors shows that the student has that factor at a high level. Cronbach's alpha value for the intrinsic motivation portion in the

TABLE 3 Content of the training achieved in experimental and control groups

Weeks	Experimental group (block-based/Scratch)	Control group (text-based/small basic)
Week 1 and 2	• Introduction to algorithm • Basic concepts • Variables • Solutions for algorithmic problems	• Introduction to algorithm • Basic concepts • Variables • Solutions for algorithmic problems
Week 3	• Algorithm examples • Introduction to programming • Control structures • Sample problem solutions	• Algorithm examples • Introduction to programming • Control structures • Sample problem solutions
Week 4	• Mathematical-logical operations • Problem solving activities	• Mathematical-logical operations • Problem solving activities
Week 5 and 6	• Condition statements with Scratch • Programming with logical operations • Problem solving activities	• Condition statements with Small Basic • Programming with logical operations • Problem solving activities
Week 7	• Program counter usage • Loops • Sample problems	• Program counter usage • Loops • Sample problems
Week 8 and 9	• Loops, arrays • Sample problems	• Loops, arrays • Sample problems
Week 10	• Number guessing game	• Number guessing game

Turkish adaptation of the scale is calculated as 0.59, and for extrinsic motivation, it is calculated as 0.63. Agreement levels with the items on the scale vary from "Absolutely incorrect for me (1)" to "Absolutely correct for me (7)" according to the seven-point Likert scale.

3.4 | Data analyses

The skewness-kurtosis and Kolmogorov–Smirnov tests were performed to decide whether the pre-analysis data had a normal distribution. In the tests performed, it was determined that all data had normal distribution. Thus, paired samples *t*-test was performed for the alteration of the variables measured in both groups from pretest to posttest; and independent samples *t*-test analyzes were performed to test whether the pretest and posttest scores differed significantly between the groups. In addition, the effect size value (Cohen's *d*) was calculated. In the article, the effect size value was interpreted as very large when above 1, large when 0.8, average when 0.5, and small (low) when 0.2 (Green & Salkind, 2005).

3.5 | Procedures

This study was conducted in "Computers-II" which is a compulsory course for the students. The Computers-II course is a 4-h course that includes programming instruction. A 10-week-long lecturing period and 2 weeks of the data collection process in the experimental and control groups were performed by the first researcher. In addition, course planning, teaching of courses, and monitoring of the participants through the whole process were performed by the first researcher. Lecturing was maintained for 10 weeks, for 4 h a week in each group. An additional 2 weeks were used to collect pretest and posttest data. In the intervention process, the experimental group was trained via block-based visual programming with Scratch and the control group was trained via text-based programming with Small Basic. Information about the topics taught in the intervention process is presented in Table 3.

Throughout the process, courses in both groups were carried out based on lecturing and problem-solving activities. In order to help with issues where students had difficulty, and to prevent out-of-purpose usage of computers, classroom management software called "Net Support School" was used. With this software, other programs installed on the computers were restricted to students' use. Question-answer sessions were carried out with students at the end of the classes every week to determine if the topic content was comprehended. The same laboratory was used for the classes of the experimental and control group and the laboratory was made suitable for programming teaching. Images related to the intervention process are presented in Figure 1.

4 | FINDINGS

4.1 | Prior checks

Before analysing the data related to the research questions, pretest scores on programming achievement, logical thinking skills, and intrinsic and extrinsic motivation were compared to evaluate the similarities of the groups. The difference between the groups in terms of pretest scores was not significant (see Table 5). The low effect size values calculated for all pretests confirmed that the groups were equivalent in terms of the variables examined.

4.2 | Findings regarding programming achievement

The differences between the pretest and posttest scores of the groups were calculated to see if there was a significant change in the achievement scores of the experimental and the control groups after the training. There was an increase in the mean achievement scores of both groups (see Figure 2) and this increase was significant for both groups (see Table 4). The calculated effect size values revealed that this increase in achievement scores was high in both groups (see Table 4).

FIGURE 1 A view from the intervention process in the experimental (left) and the control (right) groups.

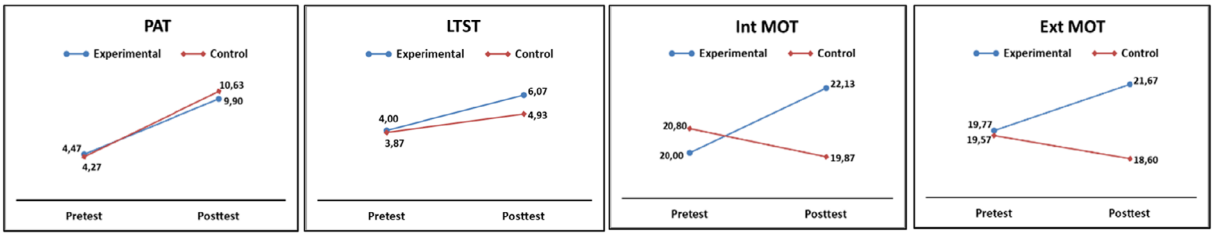

FIGURE 2 Changes in groups according to measured variables

TABLE 4 Paired samples *t*-test results regarding the variation scores of the groups from pretest to posttest

	Experimental ($N = 30$)					Control ($N = 30$)				
	Mean	SD	*t*	*p*	Cohen's *d*	Mean	SD	*t*	*p*	Cohen's *d*
PAT	5.43	2.39	12.462	0.000*	2.276	6.36	2.88	12.097	0.000*	2.209
LTST	2.07	1.72	6.578	0.000*	1.202	1.06	1.68	3.477	0.002*	0.635
Motivation (MOT)										
Int MOT	2.13	3.79	3.087	0.004*	0.564	−0.93	6.05	−0.846	0.405	−0.155
Ext MOT	1.90	4.54	2.294	0.029*	0.419	−0.97	4.96	−1.068	0.294	0.196

Note: $^*p < 0.05$.

Programming training approaches were compared in terms of their effects on programming achievement. No statistically significant difference was found ($t_{58} = -1.071$, $p > 0.05$) in terms of programming achievement between the experimental and the control groups in independent samples of *t*-tests analysis. Effect size is at a low level (Cohen's $d = 0.275$). Detailed results regarding the analysis are presented in Table 5.

4.3 | Findings regarding logical thinking skills

In this part of the study, the change and differentiation in logical thinking skill scores were examined. Depending on the training given in both groups, the logical thinking skill averages showed a significant increase (see Figure 2 and Table 4). The effect size values of this increase were high in the experimental group; it was at moderate level in the control group (see Table 4).

The difference between the groups in terms of logical thinking skills was examined, and a significant difference in the advantage of the experimental group was found in the posttest ($t_{58} = 2.159$, $p < 0.05$). The effect size shows an average level of difference (Cohen's $d = 0.557$). Results related to the analysis are presented in Table 5.

4.4 | Findings regarding intrinsic and extrinsic motivation

In the analyzes on intrinsic and extrinsic motivation scores, an increase was observed in the posttest mean scores of the experimental group, while a decrease was observed in the posttest mean scores of the control group (see Figure 2). While the increase in the scores in the experimental group was significant for both variables, the decrease in the mean scores of the control group was not significant (see Table 4). Effect size values reflected a moderate increase in the experimental group and a low-level decrease in the control group for intrinsic and extrinsic motivation scores (see Table 4).

Independent samples *t*-test was applied to test whether the intrinsic and extrinsic motivation scores showed a significant difference between the groups in the posttests. A significant difference was found in favour of the experimental group for intrinsic motivation

TABLE 5 Independent samples *t*-test results showing if there is a significant difference between the groups

		Experimental ($N = 30$)		Control ($N = 30$)				
		Mean	SD	Mean	SD	*t*	*p*	Cohen's *d*
Pretest	PAT	4.47	1.96	4.27	1.82	0.410	0.684	0.106
	LTST	4.00	2.40	3.87	2.39	0.216	0.830	0.056
	Motivation (MOT)							
	Int MOT	20.00	3.85	20.80	4.56	−0.735	0.465	0.189
	Ext MOT	19.77	5.30	19.57	5.67	0.141	0.888	0.036
Posttest	PAT	9.90	2.25	10.63	3.00	−1.071	0.288	0.275
	LTST	6.07	1.51	4.93	2.45	2.159	0.036*	0.557
	Motivation (MOT)							
	Int MOT	22.13	2.21	19.87	4.26	2.589	0.013*	0.668
	Ext MOT	21.67	2.85	18.60	4.89	2.424	0.019*	0.626

Note: *$p < 0.05$.

($t_{58} = 2.589$, $p < 0.05$), meaning the experimental group was significantly more motivated for internal factors compared to the control group. Besides, a significant difference was found in favour of the experimental group for extrinsic motivation ($t_{58} = 2.424$, $p < 0.05$), meaning the experimental group was significantly more motivated for external factors compared to the control group. An average level of effect size was found for the intrinsic (Cohen's $d = 0.668$) and the extrinsic (Cohen's $d = 0.626$) motivation. Detailed results regarding the analysis are found in Table 5.

5 | DISCUSSION

5.1 | Programming achievement

The research results show that the achievement pretest scores of both groups in which block-based visual and text-based programming training were performed increased significantly in the posttests, but this result did not show a significant difference between the groups in terms of posttest scores. The results differ from studies, which argue that achievement is higher in block-based programming environments compared to text-based programming (Hu et al., 2021; Weintrop, 2019). This emphasizes that the educational environment used in programming learning does not directly affect programming achievement. The increase in the achievement scores of the experimental group can be explained by the increased motivation levels of the students for learning programming. However, the situation is different in the control group. Although the motivation of the students in the control group decreased in the process, there was an increase in the achievement of the students. This may be due to the fact that the research was conducted with students studying in the field of mathematics. Although the students have low motivation, they can grasp the complex operations and principles of programming more easily thanks to their current mathematical knowledge and experience. In the literature, it is stated that mathematical knowledge is a significant predictor of achievement in a programming course (Ninrutsirikun et al., 2020; White & Sivitanides, 2003).

The closest studies, which are similar to this research, emphasize that achievement is higher in the Scratch environment (Cardenas-Cobo et al., 2021; Korkmaz, 2016). However, it should not be ignored that Scratch is compared with high-level programming environments in these studies. The complex nature of high-level programming environments (Java, C++, etc.) may have caused students to exert less effort. This may have caused a decrease in the achievement scores of the text-based group in these studies. In the current study, block and text-based educational environments with equivalent quality were compared. The fact that the environments that are equivalent to each other were chosen can be considered as a reason for not observing the difference in terms of achievement.

Having similar increases in the groups regarding effect size and having a lack of significant difference between the groups in the posttest mandated recalling the discussions on media and method in technology usage. Clark (1983) indicated that media do not have an effect on the training process other than being a supportive aid. He claims media choice will only affect the cost or the scope of the message intended to be transmitted, but that only the content can have a role in achievement (Clark, 1994). Kozma (1991), however, stated that there is a meaningful relation between media and method, and media influence the method used, and thus the learning process. Therefore, he argued that some students learn from media independently, while some learn by using the features of media (Kozma, 1991).

Although the results obtained from the current research verify Clark's (1983) viewpoint, Kozma's (1994) perspective regarding the statement of "Every environment is different from others by its quality" is considered important because there are some affordances provided by block-based visual programming environments in the learning process. For instance, logical processes that present some difficulties in comprehension such as variable, loop, and array can be offered in a more fun way by being embodied with the help of visual and auditory elements. In the literature, it is stated that this potential has a positive effect on the

teaching of programming (Fesakis & Serafeim, 2009; Giordano & Maiorana, 2014; Kaucic & Asic, 2011; Papadakis et al., 2016; Weintrop, 2019). In addition to these, variation in the scores of logical thinking skills and motivation can be identified as other-added values that block-based programming environments provide.

5.2 | Logical thinking skills

Another focus of the current research is the development of logical thinking skills. The training significantly affected the development of the students' logical thinking skills in the experimental and the control groups and this fact created a significant difference in the advantage of the experimental group in intergroup comparisons. According to Tuomi et al. (2018), this difference derives from the fact that logical thinking is an important precursor of problem-solving skills, which form a basis for programming. Robbins (2011) stated that logical thinking is a necessary component of problem solving. Sebetci and Aksu (2014) stated that there is a positive, average-level, and meaningful relation between students' logical thinking skills and their learning of programming.

Although the same training was given to both groups, the higher improvement in the logical thinking skills of the students in the experimental group can be explained based on several reasons. First, the fact that students were less exposed to syntax rules in Scratch may have focused their attention more on designing algorithms or structuring cause-effect relationships. The aspect of this environment that supports learning by "experimentation, experimentation and discovery" may have made students more willing to make advanced logical reasoning. It can be said that these results obtained in terms of logical thinking do not differ from the literature. For example, in Pratiwi et al.'s (2018) study, the possible reason for the higher logical thinking scores of the group trained in the Scratch environment was explained by the fact that the students had less difficulty in creating algorithms and converting the algorithm into code.

Another situation that makes the difference can be the interesting structure of the Scratch environment. It can be argued that a programming environment's interesting structure provides changes for students to spend more time in this environment, thus improving their programming experience. As Sebetci and Aksu (2014) also indicated, an increase in students' experience in programming is considered to be the reason leading to the development of logical thinking skills. Studies in the literature also show that there is a statistically significant relation between the use of a block-based visual environment and the development of logical thinking skills in programming (Costa & Miranda, 2019; Pratiwi et al., 2018; Sebetci & Aksu, 2014). In terms of logical thinking skills, the current research results support the literature.

5.3 | The intrinsic and extrinsic motivation

Another focus of the research is the changes in the motivation scores. The training activities in the experimental group significantly increased students' intrinsic motivation to learn programming. On the other hand, the intrinsic motivations of the students in the control group followed a downward trend, but this situation did not create a significant difference. In the comparison between the groups in terms of intrinsic motivation posttest scores, a significant difference was found in favour of the experimental group.

After the training, the reasons for the variation in the students' intrinsic motivation scores can be examined from three different perspectives. First, some conditions such as the use of conventional methods in programming training, the complex structure of programming that requires code writing, syntactic knowledge level errors (e.g., syntax, incorrect code writing, and bad punctuation marks) cause students' discouragement over time. This situation is considered to be a reason for the decrease in intrinsic motivation scores in the control group because motivation is affected by the environment used for teaching programming (Calder, 2010).

Second, the language of block-based programming compared to the language of text-based programming offers students the opportunity to edit the script with a language that is similar to their own language. For instance, it is expressed in accordance with the everyday language such as "increase the value of x by 1" in block-based languages whereas it is necessary to use a mathematical construct such as "$x = x + 1$" in text-based languages in order to increase the value of a variable x by 1 (Weintrop, 2019). This complex construct in text-based programming has the potential to create cognitive load (Bau et al., 2017). It can be asserted that the increase in the cognitive load of students in the control group may be a reason for the decrease in intrinsic motivation.

Third, based on the view presented by Kozma (1994) that Scratch differs in terms of an environment's nature, it is a game-based environment that allows its users to write programs using interactive stories, animations, simulations, and other dynamic media tools with the logic of a graphic block (Weintrop, 2019). In the literature, it is stated that the inclusion of multimedia elements in software is an effective method to increase the motivation of students toward learning (Brusilovsky & Spring, 2004; Kelleher et al., 2007). In addition, it is emphasized that motivation will increase in learning environments that are intriguing, amusing, and directly related to the goals (Deci et al., 2001). As a matter of fact, Scratch offers a pleasing environment to its users as they write code. Genç and Karakuş (2011) conducted a study supported by Scratch activities in order to determine the experiences and opinions of the students as a part of the course "Computer Games Design in Education". In the study, 79% of the students stated that they found the use of Scratch simple and easy, and they felt comfortable using it. In addition, the students pointed out that Scratch was an enjoyable environment and generally liked using Scratch. In addition, 73% of the students stated that learning and understanding the programming structures with Scratch was easier than other programming languages. The current research's results obtained in terms of intrinsic motivation support the existing literature.

Another crucial result of this study was the extrinsic motivation results. The training activities in the experimental group significantly increased students' extrinsic motivation. On the other hand, the

extrinsic motivations of the students in the control group followed a downward trend, but this situation did not create a significant difference. Another important issue observed in the motivation scores is that extrinsic motivation showed a significant difference in favour of the experimental group on the posttests. This situation suggests that students in the experimental group wanted to learn programming in an environment where external motivators were employed in addition to the internal tendencies such as interest and pleasure in learning, or the need for learning. This is because the increase in the extrinsic motivation of the students in the experimental group toward learning programming did not cause any negative changes in their intrinsic motivation. Three different viewpoints in the literature could be considered regarding this case.

According to the first view, the extrinsic motivation elements applied to the students to maintain their efforts do not adversely affect the students' intrinsic motivation (Cameron et al., 2001; Cameron & Pierce, 1994; Dede & Argün, 2004). This study states that the increase in the extrinsic motivation of the experimental group is caused by some potential characteristics of Scratch despite the fact that no external motivating variables were presented to the groups and the same content was processed in both groups. One of these features is that Scratch has an online interface and the written programs can be shared within this environment. Anyone in a different part of the world, who is a member of the site, has the opportunity to review the projects of others, to make comments, and to contribute to the projects they like. In the meantime, information such as how many people viewed each project, who commented on the project, and who contributed, and how much contribution was made are explained in numerical terms. This possible situation can be considered a reason for the difference in favour of the experimental group in terms of extrinsic motivation because an environment's advantage in involving basic game components such as score, number of likes, and feedback may increase the extrinsic motivation of students (Buckley & Doyle, 2016; Cozar-Gutierrez & Saez-Lopez, 2016; Hamzah et al., 2015; Papp, 2017).

According to the second view, external reinforcement applied to motivate students to learn negatively affects the intrinsic motivations of students for learning (Deci et al., 1999; Gordon, 1999; Guay et al., 2000; Hayamizu, 1997). In addition, it is stated that students who are directed to learning with the help of external reasons may experience a decrease in their achievement at the end of the process (Becker et al., 2010; Wang & Guthrie, 2004; Williams & Williams, 2011). Contrary to the second view, there appears to be a positive correlation between extrinsic motivation and achievement scores of the students in the experimental group in this study. The reason for this change has been put forward in the studies outlined in the next paragraph for articulating the third view.

In studies adopting the third view, it is stated that intrinsic and extrinsic motivation should be evaluated as complementary processes, not as processes opposed to one another (Saeed & Zyngier, 2012; Wang & Guthrie, 2004; Yıldız, 2013). Moreover, the effectiveness of intrinsic and extrinsic motivation is time- and context-oriented (Jovanovic & Matejevic, 2014; Saeed & Zyngier, 2012). Lecturers can utilize internal and external motivators at a particular time or in a specific activity (Hidi, 2000; Hidi & Harackiewicz, 2000) because the same activity can be seen as motivating by different students in intrinsic or extrinsic terms (Schunk et al., 2014). In Efecan et al.'s (2020) study, students acknowledged the real-life experience (remarkable and unremarkable stories) of experts (the ones who had higher intrinsic and/or extrinsic motivation) who had achievement in the field of programming. A significant difference was found in the intrinsic motivation of students who listened to those stories for programming. The study argued that the intrinsic or extrinsic motivation of experts in programming can be a source of intrinsic motivation for novice students.

In the explanations made within the scope of the third viewpoint, there appears to be an emphasis on context. When the teaching of programming is examined in this context, the intrinsic motivation of students can be said to be given priority, but the elements of external motivation are also important in the process because a programming course is application-oriented that involves problem solving activities in the process and requires spending a long time in front of a computer screen, and intensively engaging in cognitive skills. In such a course, students may need to be motivated together in internal and external ways to be able to maintain their efforts. Furthermore, although the programming environment used in the control group did not have any extrinsic motivation factor, students' intrinsic motivation scores decreased. For the experimental group, this suggests that the extrinsic motivators of the Scratch programming environment do not adversely affect intrinsic motivation or achievement.

5.4 | Limitations

This article is characterized by four main limitations that may inform future studies.

- Students' knowledge of math is a crucial predictor of programming achievement (Ninrutsirikun et al., 2020). Students' math scores before the study were not looked at; this can be regarded as a limitation to the current study.
- The inclusion of only the intrinsic and extrinsic dimensions of the motivation variable in the study can be stated as a limitation.
- Scratch has certain potential features (e.g., sharing programs in online environments, commenting, and the number of likes) that increase extrinsic motivation. Even though courses do not address any extrinsically motivating variables, students may make use of extrinsically motivating variables outside their courses. Therefore, the extrinsically motivating variables that can be challenging to control can be stated as study limitations.
- The fact that the experimental phase of the research was completed 5 years ago can be stated as a relative limitation. Because during this time, most countries continued to include programming teaching in their primary and secondary education curriculum (Wu et al., 2020). This situation indicates that students who will start higher education now have basic programming knowledge.

6 | CONCLUSION AND RECOMMENDATIONS

In the current study, the reflections of block-based programming training with Scratch on students' achievement, logical thinking skills, and intrinsic and extrinsic motivation scores were investigated. The results revealed a significant impact of block-based virtual programming by Scratch on logical thinking skills, intrinsic and extrinsic motivation, but the training did not have a significantly different effect on students' achievement scores. With reference to these results, some suggestions for future research are made below:

- In this study, we found evidence that block-based and text-based educational environments increase achievement, but this does not make a difference between groups. This result may be a sign that the long-standing perception of "learning programming is difficult, especially in text-based environments", may have changed. Focusing research on this issue may contribute to the improvement of the current understanding.
- Controlling the variables that predict programming achievement in future studies may contribute to more clearly revealing the results to be achieved.
- The article revealed some evidence explaining the relationship between block-based programming learning and motivation scores. In order to explain this relationship more deeply, it would be beneficial to include other components of motivation (for example, attention, confidence, satisfaction, etc.) in the process in future studies.
- Although no extrinsic motivating variable was presented to either group, the increase in extrinsic motivation in the experimental group necessitates the determination of extrinsic motivating variables in the Scratch platform and controlling these variables before the study. In future studies, it is also recommended to consider the extrinsic motivating variables of the platform.
- On the other hand, the similarities of Scratch programming language with everyday language can be an important reason for the increase in the motivation and logical thinking skills scores. Investigating this in the context of cognitive load may contribute to having a deeper understanding of the effectiveness of the programming by Scratch on motivation and logical thinking skills.
- In future studies, it is considered important to include qualitative observations in the process in order to better understand the possible reasons behind the change in programming learning and motivation scores.

ACKNOWLEDGEMENTS

This study is derived from the first author's master dissertation under the supervision of the second author. The authors would like to thank Prof. Arif Altun, Prof. Halil Yurdugül, Assoc. Prof. Hasan Çakır, Asst. Prof. Cengiz Savaş Aşkun, and anonymous reviewers for their constructive feedback.

CONFLICT OF INTEREST

The authors declare that there are no conflicts of interest associated with this publication. This manuscript is the original work of authors and all authors mutually agree for its submission.

PEER REVIEW

The peer review history for this article is available at https://publons.com/publon/10.1111/jcal.12771.

DATA AVAILABILITY STATEMENT

Not available.

ORCID

Şenol Saygıner https://orcid.org/0000-0002-5280-3847
Hakan Tüzün https://orcid.org/0000-0003-1153-5556

REFERENCES

Akbay, Ö. B. (2019). The effect of block-based programming and text-based programming environments on learning programming skills in social science education [Unpublished master's thesis]. Mimar Sinan University.

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From Scratch to "real" programming. *ACM Transactions on Computing Education (TOCE)*, *14*(4), 1–15. https://doi.org/10.1145/2677087

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the ACM*, *60*(6), 72–80. https://doi.org/10.1145/3015455

Bayman, P., & Mayer, R. E. (1983). A diagnosis of beginning programmers' misconceptions of BASIC programming statements. *Communications of the ACM*, *26*(9), 677–679. https://doi.org/10.1145/358172.358408

Baz, F. C. (2018). A comparative analysis of coding software for children. *Current Research in Education*, *4*(1), 36–47.

Becker, M., McElvany, N., & Kortenbruck, M. (2010). Intrinsic and extrinsic reading motivation as predictors of reading literacy: A longitudinal study. *Journal of Educational Psychology*, *102*(4), 773–785. https://doi.org/10.1037/a0020084

Berland, M., Martin, T., Benton, T., Petrick Smith, C., & Davis, D. (2013). Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences*, *22*(4), 564–599. https://doi.org/10.1080/10508406.2013.836655

Brusilovsky, P., & Spring, M. (2004). Adaptive, engaging, and explanatory visualization in a C programming course. Proceedings of ED-MEDIA, Lugano, Switzerland. https://sites.pitt.edu/peterb/papers/EDMED04ExVis_Final.pdf

Buckley, P., & Doyle, E. (2016). Gamification and student motivation. *Interactive Learning Environments*, *24*(6), 1162–1175. https://doi.org/10.1080/10494820.2014.964263

Büyüköztürk, Ş. (2007). *Sosyal bilimler için verianalizi el kitabı*. Pegem Yayıncılık.

Büyüköztürk, Ş., Akgün, Ö. E., Özkahveci, Ö., & Demirel, F. (2004). The validity and reliability study of the Turkish version of the motivated strategies for learning questionnaire. *Educational Sciences: Theory & Practice*, *4*(2), 207–239. https://psycnet.apa.org/record/2004-21768-001

Byrne, P., & Lyons, G. (2001). *The effect of student attributes on success in programming*. Proceedings of ITICSE, USA. https://doi.org/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. https://eric.ed.gov/?id=EJ906680

Cameron, J., Banko, K. M., & Pierce, W. D. (2001). Pervasive negative effects of rewards on intrinsic motivation: The myth continues. *The Behavior Analysis*, *24*(1), 1–44. https://doi.org/10.1007/BF03392017

Cameron, J., & Pierce, W. D. (1994). Reinforcement, reward, and intrinsic motivation: A meta-analysis. *Review of Educational Research*, *64*(3), 363–423. https://doi.org/10.2307/1170677

Cardenas-Cobo, J., Puris, A., Novoa-Hernandez, P., Parra-Jimenez, A., Moreno-Leon, J., & Benavides, D. (2021). Using Scratch to improve learning programming in college students: A positive experience from a non-weird country. *Electronics*, *10*(10), 1–15. https://doi.org/10.3390/electronics10101180

Cınar, M., Doğan, D., & Tüzün, H. (2019). Programlamaeğitimindeyapaboza öğrenmeetkinlikleri. In A. İşman, H. F. Odabaşı, & B. Akkoyunlu (Eds.), *Eğitim Teknolojileri Okumaları 2019* (pp. 505–530). Pegem Akademi.

Clark, R. E. (1983). Reconsidering research on learning from media. *Review of Educational Research*, *53*(4), 445–459. https://doi.org/10.3102/00346543053004445

Clark, R. E. (1994). Media will never influence learning. *Educational Technology Research and Development*, *42*(2), 21–29. https://doi.org/10.1007/BF02299088

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology*, *76*(6), 1051–1058. https://doi.org/10.1037/0022-0663.76.6.1051

Costa, J. M., & Miranda, G. L. (2019). Using Alice software with 4C-ID model: Effects in programming knowledge and logical reasoning. *Informatics in Education*, *18*(1), 1–15. https://doi.org/10.15388/infedu.2019.01

Cozar-Gutierrez, R., & Saez-Lopez, J. M. (2016). Game-based learning and gamification in initial teacher training in the social sciences: An experiment with MinecraftEdu. *International Journal of Educational Technology in Higher Education*, *13*(2), 1–11. https://doi.org/10.1186/s41239-016-0003-4

Deci, E. L., Kostner, R., & Ryan, R. M. (1999). A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. *Psychological Bulletin*, *125*(6), 627–668. https://doi.org/10.1037/0033-2909.125.6.627

Deci, E. L., Kostner, R., & Ryan, R. M. (2001). Extrinsic rewards and intrinsic motivation in education: Reconsidered once again. *Review of Educational Research*, *71*(1), 1–27. https://doi.org/10.3102/00346543071001001

Dede, Y., & Argün, Z. (2004). Identification of students' intrinsic and extrinsic motivation towards mathematics. *Education and Science*, *29*(134), 49–54. http://egitimvebilim.ted.org.tr/index.php/EB/article/view/5041

Deng, W., Pi, Z., Lei, W., Zhou, Q., & Zhang, W. (2020). Pencil code improves learners' computational thinking and computer learning attitude. *Computer Applications in Engineering Education*, *28*(1), 90–104. https://doi.org/10.1002/cae.22177

Denny, P., Luxton-Reilly, A., & Tempero, E. (2012). All syntax errors are not equal. In *Proceedings ofthe 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (pp. 75–80). Haifa, Israel. https://doi.org/10.1145/2325296.2325318

Denny, P., Luxton-Reilly, A., Tempero, E., & Hendrickx, J. (2011). Understanding the syntax barrier fornovices. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*. Darmstadt. https://doi.org/10.1145/1999747.1999807

Durak, H. Y. (2020). The effects of using different tools in programming teaching of secondary school students on engagement, computational thinking and reflective thinking skills for problem solving. *Technology, Knowledge and Learning*, *25*(1), 179–195. https://doi.org/10.1007/s10758-018-9391-y

Efecan, C. F., Sendag, S., & Gedik, N. (2020). Pioneers on the case for promoting motivation to teach text-based programming. *Journal of Educational Computing Research*, *1-17*, 453–469. https://doi.org/10.1177/0735633120966048

Erol, O., & Kurt, A. A. (2017). The effects of teaching programming with Scratch on pre-service information technology teachers' motivation and achievement. *Computers in Human Behavior*, *77*(2017), 11–18. https://doi.org/10.1016/j.chb.2017.08.017

Espinal, A., Vieira, C., & Guerrero-Bequis, V. (2022). Student ability and difficulties with transfer from a block-based programming language into other programming languages: A case study in Colombia. *Computer Science Education*, 1–33. https://doi.org/10.1080/08993408.2022.2079867

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. https://doi.org/10.1145/1595496.1562957

Fidai, A., Capraro, M. M., & Capraro, R. M. (2020). "Scratch"-ing computational thinking with Arduino: A meta-analysis. *Thinking Skills and Creativity*, *38*, 1–14. https://doi.org/10.1016/j.tsc.2020.100726

Fraenkel, J., Wallen, N., & Hyun, H. H. (2012). *How to design and evaluate research in education*. McGraw Hill.

Geban, Ö., Aşkar, P., & Özkan, I. (1992). Effects of computer simulations and problem-solving approaches on high school students. *Journal of Educational Research*, *86*(1), 5–10. https://www.jstor.org/stable/27540507

Genç, Z., & Karakuş, S. (2011). Tasarımla öğrenme: Eğitsel bilgisayar oyunları tasarımında Scratch kullanımı. Paper presented at the 5th International Computer & Instructional Technologies Symposium, Fırat University, Elazığ. https://web.firat.edu.tr/icits2011/icits2011ProceedingBook.pdf

Giordano, D., & Maiorana, F. (2014, April). Use of cutting edge educational tools for an initial programming course. Paper presented at the IEEE Global Engineering Education Conference (EDUCON), Military Museum and Cultural Center, Istanbul, Turkey. https://doi.org/10.1109/EDUCON.2014.6826147

Gordon, T. (1999). Çocukta dış disiplin mi? İç disiplin mi? Sistem Yayıncılık.

Green, S. B., & Salkind, N. J. (2005). *Using SPSS for Windows and Machintoch: Analyzing and understanding data*. Pearson.

Guay, F., Vallerand, R. J., & Blanchard, C. (2000). On the assessment of situational intrinsic and extrinsic motivation: The situational motivation scale (SIMS). *Motivation and Emotion*, *24*(3), 175–214. https://doi.org/10.1023/A:1005614228250

Hagge, J. (2017). Scratching beyond the surface of literacy: Programming for early adolescent gifted students. *Gifted Children Today*, *40*(3), 154–162. https://doi.org/10.1177/1076217517707233

Hamzah, A. F. W., Ali, N. H., Saman, Y. M., Yusoff, M. H., & Yacob, A. (2015). Influence of gamification on students' motivation in using e-learning applications based on the motivational design model. *International Journal of Emerging Technology in Learning*, *10*(2), 30–34. https://doi.org/10.3991/ijet.v10i2.4355

Harel, I., & Papert, S. (1991). *Constructionism*. Ablex Publishing.

Hayamizu, T. (1997). Between intrinsic and extrinsic motivation: Examination of reasons for academic study based on the theory of internalization. *Japanese Psychological Research*, *39*(2), 98–108. https://doi.org/10.1111/1468-5884.00043

Hidi, S. (2000). *An interest researcher's perspective: The effects of extrinsic and intrinsic factors on motivation*. Academic Press.

Hidi, S., & Harackiewicz, J. M. (2000). Motivating the academically unmotivated: A critical issue for the 21st century. *Review of Educational Research*, *70*(2), 151–179. https://doi.org/10.3102/00346543070002151

Hongwarittorn, N., & Krairit, D. (2010). Effects of program visualization (Jeliot3) on students' performance and attitudes towards Java programming. 8th International Conference on Computing, Communication and Control Technologies, USA. iiis.org/CDs2010/CD2010IMC/CCCT_2010/PapersPdf/TA750PM.pdf

Hu, Y., Chen, C. H., & Su, C. Y. (2021). Exploring the effectiveness and moderators of block-based visual programming on student learning: A meta-analysis. *Journal of Educational Computing Research*, *58*(8), 1467–1493. https://doi.org/10.1177/0735633120945935

ISTE. (2016). ISTE standards: Students. https://www.iste.org/standards/iste-standards-for-students

Jenkins, T. (2001). The motivation of students of programming. In Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education, Canterbury, UK. https://doi.org/10.1145/377435.377472

Jenkins, T. (2002). On the difficulty of learning to program. *Information and Computer Sciences*, *4*, 53–58. https://www.ics.heacademy.ac.uk

Jesennia, C., Puris, A., & Benavides, D. (2020). Recommender systems and Scratch: An integrated approach for enhancing computer programming learning. *IEEE Transactions on Learning Technologies*, *13*(2), 387–403. https://doi.org/10.1109/TLT.2019.2901457

Jovanovic, D., & Matejevic, M. (2014). Relationship between rewards and intrinsic motivation for learning-Research Review. *Procedia-Social and Behavioral Sciences*, *149*(2014), 456–460. https://doi.org/10.1016/j.sbspro.2014.08.287

Kandemir, C. M. (2018). Metintabanlıprogramlama. In Y. Gülbahar (Ed.), *Bilgi İşlemselDüşünmedenProgramlamaya* (pp. 265–294). Pegem Akademi.

Kandin, E., & Sendurur, E. (2022). The effects of goal-based scenarios used for programming education of 5th graders. *Interactive Learning Environments*, 1–21. https://doi.org/10.1080/10494820.2022.2036199

Karplus, R. (1977). Science teaching and the development of reasoning. *Journal of Research in Science Teaching*, *14*(2), 169–175. https://doi.org/10.1002/tea.3660140212

Kaucic, B., & Asic, T. (2011). Improving introductory programming with Scratch? Proceedings of the 34th International Convention, Opatija, Croatia. https://www.ieeexplore.ieee.org/document/5967218

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 1455–1464). San Jose, CA, USA. https://doi.org/10.1145/1240624.1240844

Kinnunen, P., & Malmi, L. (2008). CS minors in a CS1 course. In Proceedings of the Fourth International Workshop on Computing Education Research, Sydney, Australia. https://dl.acm.org/citation.cfm?id=1404529

Korkmaz, Ö. (2016). The effect of Scratch-based game activities on students' attitudes, self-efficacy and academic achievement. *Online Submission*, *8*(1), 16–23. https://doi.org/10.5815/ijmecs.2016.01.03

Kozma, R. B. (1991). Learning with media. *Review of Educational Research*, *61*(2), 179–211. https://doi.org/10.3102/00346543061002179

Kozma, R. B. (1994). Will media influence learning? Reframing the debate. *Educational Technology Research and Development*, *42*(2), 7–19. https://doi.org/10.1007/BF02299087

Lazear, D. G. (2000). *The intelligent curriculum: Using multiple intelligences to develop your students' full potential*. Zephyr Press.

Lim, G. W., & Kim, C. S. (2019). The effect of modular robot programming education on learning motivation of informatics curriculum. *The Journal of Korean Association of Computer Education*, *22*(1), 79–86. https://doi.org/10.32431/kace.2019.22.1.007

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*(2014), 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Mains, M. G. (1997). The effects of learning a programming language on logical thinking skills [Unpublished doctoral thesis]. University of Nevada.

Malik, S. I. (2018). Improvements in introductory programming course: Action research insights and outcomes. *Systemic Practice and Action Research*, *31*(6), 637–656. https://doi.org/10.1007/s11213-018-9446-y

Martinez, S. L., & Stager, G. (2013). *Invent to learn. Making, tinkering, and engineering in the classroom*. Constructing Modern Knowledge Press.

Mihci, C., & Ozdener Donmez, N. (2017). Teaching GUI-programming concepts to prospective K12 ICT teachers: MIT app inventor as an alternative to text-based languages. *International Journal of Research in Education and Science*, *3*(2), 543-559.10.21890/ijres.327912.

Mladenovic, M., Boljat, I., & Zanko, Z. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies*, *23*(4), 1483–1500. https://doi.org/10.1007/s10639-017-9673-3

Monroy-Hernandez, A., & Resnick, M. (2008). Empowering kids to create and share programmable media. *Digital Library*, *15*(2), 50–53. https://doi.org/10.1145/1340961.1340974

Moors, L., Luxton-Reilly, A., & Denny, P. (2018). Transitioning from block-based to text-based programming languages. In *2018 international conference on learning and teaching in computing and engineering*. (pp. 57–64). Auckland, New Zealand. https://doi.org/10.1109/LaTICE.2018.000-5

Ninrutsirikun, U., Imai, H., Watanapa, B., & Arpnikanondt, C. (2020). Principal component clustered factors for determining study performance in computer programming class. *Wireless Personal Communications*, *1-20*, 2897–2916. https://doi.org/10.1007/s11277-020-07194-5

Noone, M., & Mooney, A. (2018). Visual and textual programming languages: A systematic review of the literature. *Journal of Computers in Education*, *5*(2), 149–174. https://doi.org/10.1007/s40692-018-0101-5

Nouri, J., Zhang, L., Mannila, L., & Noren, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*, *11*(1), 1–17. https://doi.org/10.1080/20004508.2019.1627844

Oliva, J. M. (2003). The structural coherence of students' conceptions in mechanics and conceptual change. *International Journal of Science Education*, *25*(5), 539–561. https://doi.org/10.1080/09500690210163242

Paas, F., Renkl, A., & Sweller, J. (2004). Cognitive load theory: Instructional implications of the interaction between information structures and cognitive architecture. *Instructional Science*, *32*(1), 1–8. https://doi.org/10.1023/B:TRUC.0000021806.17516.d0

Papadakis, S., & Kalogiannakis, M. (2019). Evaluating a course for teaching introductory programming with Scratch to pre-service kindergarten teachers. *International Journal of Technology Enhanced Learning*, *11*(3), 231–246. https://doi.org/10.1504/IJTEL.2019.10020447

Papadakis, S., Kalogiannakis, M., & Zaranis, N. (2016). Developing fundamental programming concepts and computational thinking with Scratch Jr in preschool education: A case study. *International Journal of Mobile Learning and Organisation*, *10*(3), 187–202. https://doi.org/10.1504/IJMLO.2016.077867

Papert, S. (2013). *Teaching children thinking*. Psychology Press.

Papert, S., & Solomon, C. (1971). Twenty things to do with a computer. *Educational Technology Magazine*, *248*(3), 1–40.

Papp, T. A. (2017). Gamification effects on motivation and learning: Application to primary and college students. *International Journal for Cross-Disciplinary Subjects in Education*, *8*(3), 3193–3201. https://doi.org/10.20533/ijcdse.2042.6364.2017.0428

Pintrich, P. R. (2003). A motivational science perspective on the role of student motivation in learning and teaching contexts. *Journal of Educational Psychology*, *95*(4), 667–686. https://doi.org/10.1037/0022-0663.95.4.667

Pintrich, P. R., Smith, D. A., Garcia, T., & McKeachie, W. J. (1991). *A manual for the use of the motivated strategies for learning questionnaire (MSLQ)*. University of Michigan.

Pratiwi, U., Sriyono, S., & Akhdinirwanto, R. W. (2018). Student computational logical thinking of block programming concept in Arduino learning by S4A (Scratch for Arduino). *Advances in Social Science, Education*

and Humanities Research, *2018*(12), 245–248. https://doi.org/10.2991/aecon-18.2018.47

Quahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A., & Lahmine, S. (2015). Learning basic programming concepts by creating games with Scratch programming environment. *Procedia-Social and Behavioral Sciences*, *191*, 1479–1482. https://doi.org/10.1016/j.sbspro.2015.04.224

Quille, K., & Bergin, S. (2019). CS1: How will they do? How can we help? A decade of research and practice. *Computer Science Education*, *29*(2–3), 254–282. https://doi.org/10.1080/08993408.2019.1612679

Resnick, M. (2012). Reviving Papert's dream. *Educational Technology*, *52*(4), 42–46. https://www.jstor.org/stable/44430058

Robbins, J. K. (2011). Problem solving, reasoning, and analytical thinking in a classroom environment. *The Behavior Analyst Today*, *12*(1), 41–48. https://eric.ed.gov/?id=EJ958875

Ruf, A., Mühling, A., & Hubwieser, P. (2014). Scratch vs. Karel: Impact on learning outcomes and motivation. In the Proceedings of the 9th Workshop in Primary and Secondary Computing Education, Berlin, Germany. https://doi.org/10.1145/2670757.2670772

Ryan, R. M., & Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, *25*(1), 54–67. https://doi.org/10.1006/ceps.1999.1020

Saeed, S., & Zyngier, D. (2012). How motivation influences student engagement: A qualitative case study. *Journal of Education and Learning*, *1*(2), 252–267. https://doi.org/10.5539/jel.v1n2p252

Saygıner, Ş., & Tüzün, H. (2018). Programlama eğitimi üzerine bir inceleme: Yaşanan zorluklar, mevcut uygulamalar ve güncel yaklaşımlar. In B. Akkoyunlu, A. İşman, & H. F. Odabaşı (Eds.), *Eğitim Teknolojileri Okumaları 2018* (pp. 693–710). Pegem Akademi.

Schunk, D., Meece, P., & Pitrinch, P. (2014). *Motivation in education: Theory, research, and applications*. Pearson Education.

Sebetci, O., & Aksu, G. (2014). The effect of logical and analytical thinking skills on computer programing languages. *Educational Sciences and Practice*, *13*(25), 65–83. http://ebuline.com/pdfs/25Sayi/25.pdf#page=73

Seidman, R. H. (1981). *The effects of learning a computer programming language on the logical reasoning of school children*. American Educational Research Association.

Siegel, E. V. (1999). Why do fools fall into infinite loops: Singing to your computer science class. *ACM SIGCSE Bulletin*, *31*(3), 167–170. https://doi.org/10.1145/384267.305909

Souza, S. M., & Bittencourt, R. A. (2019). Motivation and engagement with PBL in an introductory programming course. In *Proc. IEEE Front. Educ. Conf. (FIE)* (pp. 1–9). Covington, KY, USA. https://doi.org/10.1109/FIE43999.2019.9028419

Swain, P. E. (2013). Raptor a vehicle to enhance logical thinking. *Journal of Environmental Hazards*, *7*(4), 353–359. https://peer.asee.org/raptor-a-vehicle-to-enhance-logical-thinking.pdf

Tijani, F., Callaghan, R., & De Villers, R. (2020). An investigation into pre-service teachers' experiences while transitioning from Scratch programming to procedural programming. *African Journal of Research in Mathematics, Science and Technology Education*, *24*(2), 266–278. https://doi.org/10.1080/18117295.2020.1820798

Tobin, K. G., & Capie, W. (1981). The development and validation of a group test of logical thinking. *Educational and Psychological Measurement*, *41*(2), 413–423. https://doi.org/10.1177/001316448104100220

Topallı, D., & Çağıltay, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers & Education*, *120*(2018), 64–74. https://doi.org/10.1016/j.compedu.2018.01.011

Tuomi, P., Multisilta, J., Saarikoski, P., & Suominen, J. (2018). Coding skills as a success factor for a society. *Education and Information Technologies*, *23*(1), 419–434. https://doi.org/10.1007/s10639-017-9611-4

Wang, J. H., & Guthrie, J. T. (2004). Modeling the effects of intrinsic motivation, extrinsic motivation, amount of reading, and past reading achievement on text comprehension between U.S. and Chinese students. *Reading Research Quarterly*, *39*(2), 162–186. https://doi.org/10.1598/RRQ.39.2.2

Weintrop, D. (2019). Block-based programming in computer science education. *Communications of the ACM*, *62*(8), 22–25. https://doi.org/10.1145/3341221

Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education*, *142*(2019), 1–17. https://doi.org/10.1016/j.compedu.2019.103646

White, G., & Sivitanides, M. (2003). An empirical investigation of the relationship between success in mathematics and visual programming courses. *Journal of Information Systems Education*, *14*(4), 409–416. http://jise.org/volume14/n4/JISEv14n4p409.html

Williams, K. C., & Williams, C. C. (2011). Five key ingredients for improving student motivation. *Research in Higher Education Journal*, *11*, 1–23. http://aabri.com/manuscripts/11834.pdf

Wu, L., Looi, C. K., Multisilta, J., How, M. L., Choi, H., Hsu, T. C., & Tuomi, P. (2020). Teacher's perceptions and readiness to teach coding skills: A comparative study between Finland, mainland China, Singapore, Taiwan, and South Korea. *The Asia-Pacific Education Researcher*, *29*(1), 21–34. https://doi.org/10.1007/s40299-019-00485-x

Xu, Z., Ritzhaupt, A. D., Tian, F., & Umapathy, K. (2019). Block-based versus text-based programming environments on novice student learning outcomes: A meta-analysis study. *Computer Science Education*, *29*(2–3), 177–204. https://doi.org/10.1080/08993408.2019.1565233

Yıldız, M. (2013). The role of the reading motivation, reading fluency and reading comprehension on Turkish 5th graders' academic achievement. *Turkish Studies*, *8*(4), 1461–1478. https://doi.org/10.7827/TurkishStudies.4780

Yusoff, K. M., Ashaari, N. S., Wook, T. S. M. T., & Ali, N. M. (2020). Analysis on the requirements of computational thinking skills to overcome the difficulties in learning programming. *International Journal of Advanced Computer Science and Applications*, *11*(3), 244–253. https://doi.org/10.14569/IJACSA.2020.0110329

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education*, *141*(2019), 1–25. https://doi.org/10.1016/j.compedu.2019.103607

How to cite this article: Saygıner, Ş., & Tüzün, H. (2023). The effects of block-based visual and text-based programming training on students' achievement, logical thinking skills, and motivation. *Journal of Computer Assisted Learning*, *39*(2), 644–658. https://doi.org/10.1111/jcal.12771