Journal of Modern Technology and Engineering
Vol.10, No.1, 2025, pp.26-41
https://doi.org/10.62476/jmte.10126

THE EFFECT OF TOXICODE AND GOOGLE BLOCKLY APPLICATIONS ON 5TH GRADE STUDENTS' ALGORITHM DEVELOPMENT AND COMPUTATIONAL THINKING SKILLS [1]

Bülent Koçak[1*], **Mesut Türk**[2]

[1]Ministry of National Education, Ankara, Turkiye
[2]Amasya University, Amasya, Türkiye

Abstract. This study aims to investigate the effects of Toxicode and Google Blockly applications on 5th grade students' computational thinking and algorithm development skills. The study included 102 students, with 54 in the experimental groups and 48 in the control groups, following the Solomon four-group model. In the experimental groups, Toxicode program activities were applied; in the control groups, the activities of the Google Blockly application in the textbook of the Ministry of National Education were applied. While post-test was applied to all groups in the experimental process, pre-test was applied only to the experimental1 and control1 groups. In the data collection process, "Computer Thinking Scale" and "Algorithm Development Self-Efficacy Perception Scale" were used. The results show that Toxicode and Google Blockly applications have a positive effect on both skills. While students' self-efficacy perception of algorithm development and computational thinking skills increased, it was concluded that this increase was statistically significant only in the self-efficacy perception of algorithm development in favor of the control group. As a result, block-based and visual programming tools contribute to the development of various skills such as computational thinking and algorithm development by making learning programming easier and more fun.

Keywords: algorithm development, blockly, computational thinking, toxicode.
Corresponding author: Bülent, Koçak, Ministry of National Education, Ankara, Türkiye, Tel.: +90(312) 413 13 27, e-mail: *bulent.kocak@meb.gov.tr*
Received: 22 August 2024; Revised: 3 December 2024; Accepted: 4 January 2025; Published: 30 April 2025.

1 Introduction

One of the objectives of education is to ensure that the new generation can adapt to rapidly developing technology and contribute to this rapid technological development. People who are competent in technological literacy, have developed skills in problem solving, can use their creativity in all areas of life, and have computational thinking skills will be able to keep up with this new and rapidly changing era (Özel, 2019). Computational thinking and coding are essential skills that enable a young person to combine his/her creativity with his/her talents and produce a product to be part of productive society.

While the concept of computational thinking has been gaining prominence in recent years, Papert (1980) argued that computational thinking skills can contribute to learners' affective development by supporting individuals' cognitive development, learning, and thinking skills. Basically, computational thinking is a type of analytical thinking that individuals use to solve

[1]This study is derived from the first author's master's thesis conducted under the supervision of the second author.

problems (Wing, 2008). According to Futschek (2006), the concept of computational thinking consists of sub-skills such as analyzing, understanding, expressing a problem, developing suggestions for solving the problem, and creating algorithms for the solutions developed.

Each country's educational policy on computational thinking and coding may differ. When the curricula in Türkiye are examined, it is seen that some of the subcomponents of students' computational thinking skills are already included in the curriculum when the fifth grade Information Technologies and Software course (ITSC) curriculum is examined (MoNE, 2018). In addition, knowing that students have the necessary skills in this subject and eliminating the deficiencies has become a necessity rather than a preference today (Barr et al., 2011).

With computational thinking, students can understand the process of learning, understand the meaning of problems, and build algorithms for solving those problems. In this context, problem solving, algorithm, algorithmic thinking, logical reasoning, abstraction, and decomposition are among the concepts included in the definition of computational thinking skills (Selby, 2014). Denning (2003), defines the concept of computational thinking from different perspectives as a process involving the input of information and the creation and implementation of an algorithm. The algorithm is the name of the process steps to achieve the target goal (Aytekin et al., 2018). To achieve the result, solution methods should be specified in detail in the created algorithms. In fact, it is not the program that determines the steps to be taken when solving a problem on a computer or in life, but the method that determines how the program should be. In other words, an algorithm is a set of methods designed to solve a problem (Sedgewick & Wayne, 2011).

Software development (programming) is a process that includes all the steps from the design stage to implementation and higher order thinking skills such as algorithmic thinking, critical thinking, creative thinking, reflective thinking and computational thinking (Akçay & Çoklar, 2016; Brennan & Resnick, 2012; Duncan et al., 2014; Fessakis et al., 2019; Morelli et al., 2011). It is known that there is a positive relationship between programming and computational thinking skills (Oluk & Korkmaz, 2016). However, programming involves the process of algorithm development and algorithm development requires computational thinking. A programmer uses computational thinking skills to analyze a problem, create an appropriate algorithm, organize the steps and evaluate the results. Therefore, programming education and practices help students both develop their programming skills and strengthen their computational thinking skills. Programming activities allow students to develop algorithmic thinking skills, reflective thinking skills, critical thinking skills, problem solving strategies and logical reasoning skills (Akçay & Çoklar, 2016; Duncan et al., 2014).

Providing programming education to children at an early age will increase their self-confidence and ensure that a skill learned early will be stronger and more applicable in the cognitive process (Kert & Uğraş, 2009). Children who encounter computational thinking and coding for the first time at school age may experience faltering. The most important reason for this is that they have difficulty in understanding the logic of coding (Çağıltay & Fal, 2014). For this reason, in addition to the subject to be taught, how and with which methods that education will be given has also gained importance. Traditional programming languages have certain syntax rules. However, visual or block-based applications provide the opportunity to code without the need for such rules (Kraleva et al., 2019). Thanks to the advantages of visual and block-based coding, coding education has become easier even for young and novice learners. Students develop a positive attitude towards programming with visual and block-based coding without experiencing the difficulties they encounter in text-based programming languages (Alp, 2019).

At the kindergarten and primary school level, it would be better to use block-based programming tools that can work in a drag-and-drop manner and whose errors can be corrected more easily instead of abstract and difficult to understand codes Resnick et al. (2009). Alice, Bitsbox, Codecombat, Microsoft Small Basic, Scratch, Lightbot, KoduGame Lab, Stagecast Creator, ToonTalk, App Inventor, CodeAcademy, KhanAcademy, Code.org, etc. are the main environments where block-based programming can be realized (Korkmaz et al., 2021).

The reason for choosing a block-based programming environment for early learners is to minimize the difficulties encountered in text-based coding, the difficulty of understanding the abstract structure, and to keep the interest of the learners at the highest level until then. It is seen that all of these applications have a common goal. This goal is to answer the question "How can programming be taught in the most understandable, efficient, easiest and charming way with user-friendly interface?".

1.1 Literature Review

The relationship between programming, computational thinking, and algorithm development is quite strong (Grover & Pea, 2013). There is a strong positive correlation between computational thinking, and algorithm development skills required to be competent in programming (Korkmaz et al., 2017). While algorithmic thinking strategies are used in the programming process, computational thinking skills also come into play and enable effective management of the problem solving process (Guadalupe & Fumarola, 2015). Various studies show that programming tools support students' development of algorithmic thinking (Fessakis et al., 2019), creative thinking (Morelli et al., 2011) and computational thinking (Brennan & Resnick, 2012; Fessakis et al., 2019), which are considered as higher-order thinking skills. Programming activities support students' analytical thinking skills, problem-solving strategies, logical reasoning skills, and creativity (Akçay & Çoklar, 2016; Duncan et al., 2014). Many of these skills are considered 21st century skills and today's learners are expected to have these skills in the near future (BattelleforKids, 2023; Sayın & Seferoglu, 2016). Without being bound to a specific programming language (Futschek, 2006), or even with the computer science without computers movement (Henderson, 2008), activities can be organized for the development of students' algorithm and computational thinking skills starting from a young age. At this point, the teaching approach and the applications used gain importance.

Block-based tools are one of the most effective teaching tools used to teach programming, especially to younger age groups. Students use their problem solving, logical thinking, and algorithm building skills by selecting and combining the necessary blocks to solve a specific problem. This process helps them develop the thought processes required in computer science and especially in programming (Kalelioglu & Gülbahar, 2014). Block-based programming platforms provide students with the opportunity to become familiar with basic programming concepts before learning complex coding languages, making it easier for students to learn more complex languages in the future and increasing their perception of self-efficacy in programming (Fraser, 2015).

According to Batni et al. (2025), block-based programming platforms represent a significant advancement in K-12 education, provide an innovative and accessible way to meet programming concepts. A systematic review study, conducted by Sonjaya & Munir (2025), showed that using block-based programming platforms could improve basic programming understanding, motivation, creativity, problem-solving skills and self-efficacy of students regarding programming concepts. According to their results, the block-based programming teaching approach has a positive impact on students' computational skills development.In their study, Deng et al. (2020) state that block-based programming (such as Pencil Code) is an important part of developing students' computational thinking skills at the k-12 education level. Rovshenov & Sarsar (2023), in a trend study of 162 articles indexed in the SSCI index between 2012 and 2020, found that programming education has a positive effect on students' computational thinking skills, learning, and academic achievement.

Oluk et al. (2018)'s (2018) research examines the effect of Scratch programming tool on 5th grade students' algorithm development and computational thinking skills. Their study was conducted with a quasi-experimental design with a pre-test&post-test control group, with a total of 62 students, 31 in the experimental group and 31 in the control group. In their intervention process, the students in the experimental group were taught the algorithm using

the Scratch program, while the students in the control group were taught the algorithm in line with the existing curriculum. According to the pre-test and post-test results, students' algorithm development skills and computational thinking skills increased significantly in favor of the experimental group. The study of Oluk et al. (2018) shows that the Scratch program is an effective learning tool in developing algorithm development and computational thinking skills.

Deniz (2021) evaluates the effect of Tinkercad on students' computational thinking skills and perceptions in programming education in her master thesis. The sample group of the her study consisted of 583 middle school students studying at the 5th, 6th, 7th, and 8th grade level in the 2019-2020 academic year and 11 information technologies teachers. In her study, it was found that students' perceptions of computational thinking were at a medium level, there was a low positive relationship between the frequency of Tinkercad use and Tinkercad perceptions, while there was a positive relationship between Tinkercad perceptions and computational thinking skills.

When programming education in Türkiye is examined, it is seen that online block-based coding tools such as Google Blockly, Scratch and code.org are frequently preferred in the Information Technologies and Software course at the primary education level (Dönmez Usta & Turan Güntepe, 2019; Yüksel, 2017). Commonly used block-based coding platforms (code.org, Google Blockly, etc.), users are expected to use the code blocks required to fulfill the tasks in the activities, whereas in the Toxicode platform, users are expected to complete the task according to the statements given to represent the code blocks. The difference between "completing tasks using code blocks" and "completing tasks based on statements given to represent code blocks" targets different skill sets in the learning process and provides learners with different types of computational and cognitive abilities. In this context, it is seen that there are not enough studies investigating how Toxicode-like platforms have an effect on students' computational thinking and algorithm development self-efficacy perceptions (Kalelioglu & Gülbahar, 2014; Yükseltürk & Altıok, 2015). The purpose of this study is to comparatively examine the effect of activities in Toxicode and Google Blockly applications on 5th grade students' computational thinking skills and algorithm development self-efficacy perceptions in information technologies and software course.

2 Method

This section describes the research steps, including the selection of the research design, participants, experimental process, data collection tools, and data analysis.

2.1 Research Design

In this study, Solomon four-group model, one of the experimental designs, was used to examine the effects of Toxicode and Google Blockly applications on students' computational thinking skills and self-efficacy perception of algorithm development. This model is based on a quasi-experimental structure in which control groups and experimental groups are randomly selected. In this model, there are 2 experimental groups, one of them is administered a pre-test while the other is not. Similarly, one of the 2 control groups is administered a pre-test while the other is not. At the end of the process, a post-test is applied to all groups. Experiment_1 and Control_1 groups represent the pretest-posttest control group model, while Experiment_2 and Control_2 groups represent the posttest control group model (Fraenkel & Wallen, 2008). Figure 1 shows the Solomon four-group research design which was used in the study.

The Solomon Four Group Design is the most robust experimental model that preserves both internal and external validity (Karasar, 2012). It is especially used to control measurement, timing and maturity effects that may affect internal validity and to increase the generalizability of the results (Kirk, 2009). This model provides the opportunity to test the effect of pretest on

Group	Pre-test	Intervention	Post-test
Experimental_1	CTS + ADSES	Toxicode	CTS + ADSES
Control_1	CTS + ADSES	-	CTS + ADSES
Experimental_2	-	Toxicode	CTS + ADSES
Control_2	-	-	CTS + ADSES

ADSES: Algorithm Development Self-Efficacy Scale; CTS: Computational Thinking Scale

Figure 1: Solomon 4 group research design

Group	Classes	Intervention Method	Girls n	Boys n	Total n	%
Experimental_1	5-D	Toxicode	13	13	26	25,5
Control_1	5-F	Blockly	15	12	27	26,5
Experimental _2	5-M	Toxicode	10	18	28	27,5
Control_2	5-B	Blockly	10	11	21	20,6
		Total	48	54	102	100

Figure 2: Participants

the change in posttest scores, which is called "pretest sensitivity". By increasing internal and external validity, it allows researchers to obtain more reliable and valid results. In addition, its applicability and practicality make it possible to form experimental groups and plan the data collection process more flexibly (Campbell & Stanley, 1963).

2.2 Participants

The study group of the research consisted of a total of 102 students studying in the 5th grade at a public school in a city located in the North of Türkiye in the 2022-2023 academic year (see Figure 2). Students were not true randomly assigned to the experiment and control groups in order to avoid disrupting their other courses' flows at school and to avoid problems that may arise from the change in classroom dynamics when a new class is created. The class sections in which the students were currently studying were randomly selected as experiment and control groups. Therefore, random assignment was based on students' class sections instead of students. Due to this distinction in random assignment, the study was designed as quasi-experimental rather than truly experimental. In this quasi-experimental research, experiment and control groups were randomly assigned according to class sections. Convenient sampling method was utilized to specify the participants in terms of easy to reach and make the research process feasible (Yıldırım & Şimşek, 2011). The utilization of convenient sampling, favored by researchers when reaching the entire population is impractical or when a significant time investment is required, could be regarded as a study's constraint. This is particularly pertinent when considering the necessity for participant coordination, the approval process, and substantial research financial resources.

2.3 Data Collection Tools

The data were collected using two scales namely, Computational Thinking Scale and Algorithm Development Self-Efficacy Perception Scale. Before using the scales, permission to use the scales was obtained from the authors who developed them.

2.3.1 Computational Thinking Scale(CTS)

The CTS which was developed by Korkmaz et al. (2015) was used to understand the computational thinking skill levels of middle school students. This scale is a 22-item Likert-type (5-point) scale based on 5 factors: creative thinking, critical thinking, algorithmic thinking, collaborative thinking, and problem solving. As a result of the factor analysis conducted by the authors during the development of the scale, the correlation coefficient value range of the items ranged from 0.6555 to 0.862, and the regression value range ranged from 0.507 to 0.872; Cronbach Alpha reliability coefficient was 0.809. (Korkmaz et al., 2015).

2.3.2 Algorithm Development Self-Efficacy Perception Scale(ADSES)

ADSES was developed by Bakırcı (2019) to evaluate middle school students' self-efficacy perceptions of algorithm development. In the 16-item scale study, it was seen that a one-factor structure explaining 91.929% of the total variance in original scale's factor analysis. In addition, the internal consistency coefficient of the scale was found as 0.994 according to Cronbach's Alpha analysis and the Bakırcı (2019) stated that the scale is a valid and reliable scale that can be used in the literature.

2.4 Experimental Process

The ethics committee and MoNE application permissions required to conduct the application in 5th grade students' information technology and software course before the experimental processes. The research was conducted with four different 5th grade classes and an information technologies teacher within a four-week period. The students who were absent during the experimental implementation process and did not actively participate in the process, and their data were excluded from the data set and were not considered in the analysis.

For the experimental group, activities were designed on the Toxicode platform to increase students' computational thinking skills and their self-efficacy in algorithm development. For these activities, two field experts were consulted and the activities were finalized. In the control group, lesson plans for the Google Blockly application in the existing curriculum and teacher's guidebook were prepared and implemented. In the experimental and control groups, the teaching process was carried out for 4 weeks through activities in accordance with the objectives of the course. Partial screenshots of the sample course plan for the Toxicode and Google Blockly applications are given in Figure 3 and the applications' interfaces in Figure 4

2.5 Data Analysis

At the end of the experimental process, the meta-analytic approach proposed by Braver & Braver (1988) was utilized in the comparisons between the experimental and control groups. The most important aspect of this approach is that it makes it possible to evaluate the effects of the experimental process, the results of the groups receiving the pre-test and the results of the groups receiving the post-test. The data analysis flow diagram specific to this model is presented in Figure 5.

The number of participants in each of the groups in the study meets the minimum number of participants required for experimental research. According to Cohen et al. (2007), this number should be 15 and above. The number of participants in the groups in the study were 21,

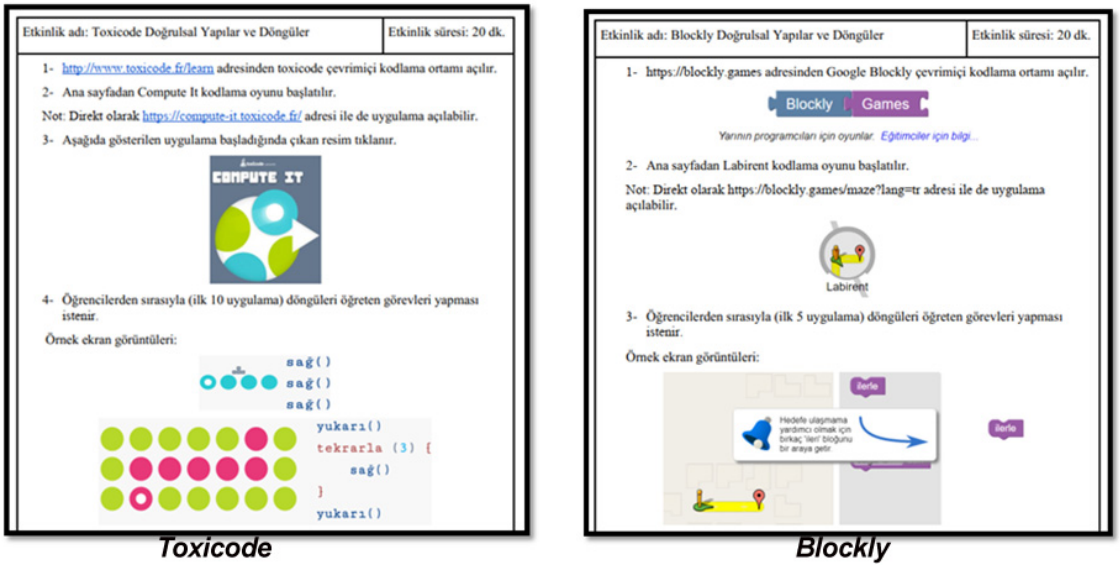

Figure 3: Sample course plans in Toxide and Blockly

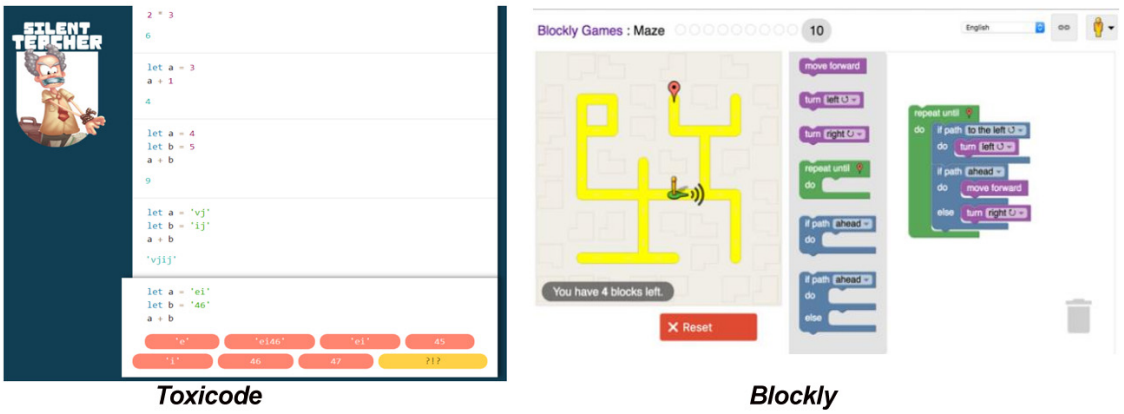

Figure 4: Toxicode and Blockly applications' interfaces

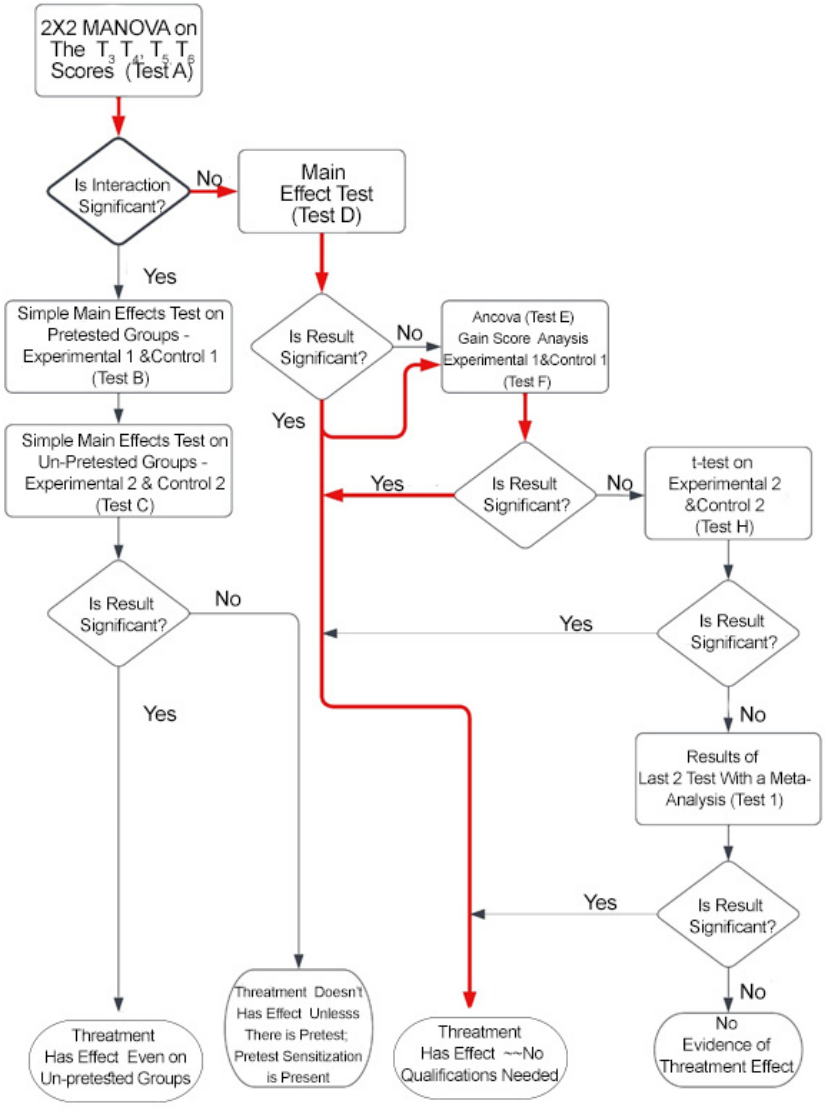

Figure 5: Data Analysis Process Flowchart (Adopted from Braver & Braver (1988))

26, 27 and 28. After the data collected within the scope of the study were entered into the SPSS package program, missing data and outliers were checked. The data set was finalized, and normality tests were performed. In order to perform parametric tests, the criterion that kurtosis and skewness values should be between -1.5 and +1.5 (Tabachnick & Fidell, 2013) was taken into consideration. In addition the skewness and kurtosis values, Kolmogorov-Smirnov normality test, histograms and box-plots, were examined to determine whether the pre-test and post-test data of the CTS and ADSES scales, were normally distributed. In the Kolmogorov-Smirnov test, data sets with a significance level above .05 are considered to be normally distributed (Özdamar, 2013). Considering these criteria, it was seen that the pre-test data of the CTS and ADSES scales of the Experimental-1 and Control-1 groups, which were pretested in the study, were above .05, which is accepted as the significance level for the Kolmogorov-Smirnov test. The skewness and kurtosis values of the same data set were found to vary between -1,018 and -,017. According to the results of both examinations, it was concluded that the data belonging to the pretest scores of the CTS and ADSES scales were normally distributed. The results of the analysis of the CTS and ADSES post-test data of all experimental and control groups are above the significance level of .05 of the Kolmogorov-Smirnov test. Skewness and kurtosis values of the same data set were found to vary between -,367 and ,149. According to these results, which are within the range of acceptable values, it was concluded that the data exhibited a normal distribution in the analysis results of the post-test data of CTS and ADSES.

Subsequently, MANOVA (Test A), which is a multivariate analysis of variance method that allows the dependent variables to be analyzed at the same time, was performed. Afterwards, since there was no statistically significant difference, "Main Effect Analysis (Test D)" was performed to look at the effect of the method. This analysis allowed comparisons to be made when evaluating the effects of the experimental and control groups on the experimental procedure, regardless of whether the groups were pre-tested or not. According to the results of this comparison, it was revealed that the method had a statistically significant effect. By controlling the pre-test data in the groups to which the pre-test was applied, the effect of the experiment on the post-test scores is tried to be revealed (Büyüköztürk, 2007). For this purpose, covariance analysis method, ANCOVA, was conducted to reveal the difference between Experiment_1 and Control_1 groups. Since statistically significant results were obtained because of this analysis, independent sample t-test was applied to the pre-test and post-test score differences of the two dependent variables in order to see the change between the groups.

3 Results

After the intervention process, the results of the Algorithm Development Self-Efficacy Perception Scale (ADSES) and Computational Thinking Scale (CTS) were analyzed by group based, in terms of experimental or control groups. The results of the these scales and descriptive statistics for groups are given in the Table 1.

Table 1 shows that for the ADSES, the group with the highest pre-test mean was Control-1 ($\bar{X}$ =3.10), while the group with the highest post-test mean was also Control-1 ($\bar{X}$ =3.80). For the CTS, the highest pre-test mean was Control-1 ($\bar{X}$ =3.32), while the highest post-test mean was Control-1 ($\bar{X}$ =3.86) either.

In order to see the equivalence of the Experiment-1 and Control-1 groups before the experimental application, independent samples t-test was performed for both scales. It was concluded that there was no significant difference between the groups for both ADSES (t(99)= -1,73, p>.05) and CTS (t(99)= -1,84, p>.05). Therefore, it can be said that the groups had similar characteristics at the beginning. After determining the equivalence of the groups, in the first step of the analysis, it was tried to test whether there was a pre-test sensitivity. For this purpose, a MANOVA (2x2) analysis was performed for the post-test scores of all groups, and it was examined whether the post-test scores of all groups changed depending on the pre-test.

Table 1: Results of ADSES and CTS Scales/Tests for Research Groups

Scale/Test	**Time**	**Group**	**n**	**Mean ($\bar{X}$)**	**Sd.**
ADSES	Pre-test	Experimental-1	26	2.94	0.29
		Control-1	27	3.10	0.34
	Post-test	Experimental-1	26	3.40	0.38
		Control-1	27	3.80	0.54
		Experimental-1	28	3.44	0.74
		Control-1	21	3.63	0.52
CTS	Pre-test	Experimental-1	26	3.10	0.43
		Control-1	27	3.32	0.47
	Post-test	Experimental-1	26	3.49	0.43
		Control-1	27	3.86	0.61
		Experimental-1	28	3.47	0.70
		Control-1	21	3.61	0.43

Table 2: MANOVA (2x2) results for the post-test scores of CTS & ADSES scales according to the pre-test independent variable

Source	Wilks' λ	F	df1	df2	p	η_p^2
Pretested & Un-pretested	0.984	1.08	3	225	0.441	0.016

According to the MANOVA result in Table 2 there was no statistically significant difference between the groups according to the dependent composite variable formed from the post-test scores (F(3, 225) = 1.08; p>.05; Wilks'λ= ,984; η_p^2= ,016). With this finding, it was seen that the group averages obtained from the CTS and ADSES post-test scores did not change depending on the pre-test scores. Therefore, based on the analyses conducted in this experimental study, it was concluded that pre-test sensitivity was not present.

Table 3: MANOVA (2x2) results for the post-test scores of CTS & ADSES scales according to method independent variable

Source	Wilks' λ	F	df1	df2	p	η_p^2
Method (Toxicode - Blockly)	0.922	1.12	3	329	0.018	0.078

Table 3 shows the data belonging to the method independent variable of the MANOVA (2X2) analysis performed on the post-test scores of the groups. When Table 3 is analysed, a statistically significant difference was observed between the groups according to the dependent composite variable formed over the post-test scores (F(3, 329) = 1,12; p<.05; Wilks'λ= ,922; η_p^2= ,078). This result shows that the group mean scores formed from the CTS and ADSES post-test scores changed depending on the intervention method. As a result of these analyses, it was found that students' computational thinking skills and algorithm development self-efficacy perceptions increased significantly in the ITSC course conducted using the applications on the Toxicode online platform. Figure 6 summarizes the changes in the pre-test and post-test results of the scales depending on the method (intervention) applied.

Although there was a difference in the ADSES post-test scores according to intervention method in groups depending on the method independent variable of the ADSES post-test scores ($\bar{X}$Toxicode= 3,42; $\bar{X}$Google_Blockly=3,73; p<,05); according to the results of MANOVA analysis, this difference was found to be statistically significant. In addition, while there was a difference in the CTS post-test scores according to applied intervention method in groups in tems of intervention method as independent variable ($\bar{X}$Toxicode= 3,48; $\bar{X}$Google_Blockly=3,75; p<,05); according to the results of MANOVA analysis, this difference was found to be statistically significant. Depending on the next step of the meta-analytic (Braver & Braver, 1988)

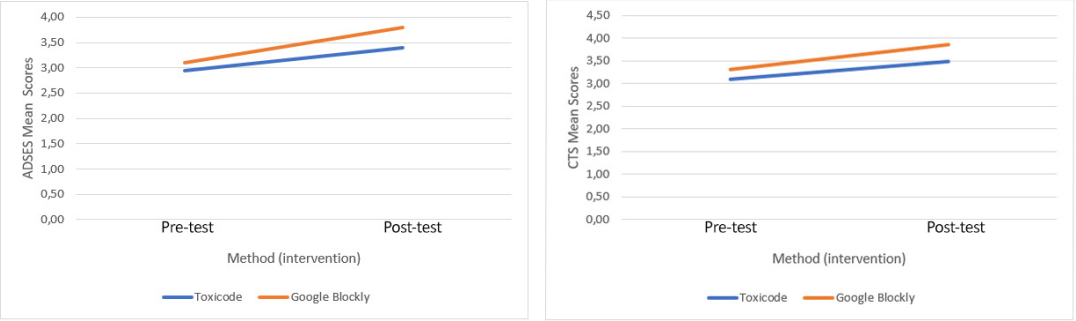

Figure 6: Groups' changing mean scores of CTS and ADSES scores by intervention method

approach, ANCOVA analysis (see table 4) was performed between the scores generated by controlling the pre-test scores and taking the post-test mean score differences of all groups to determine the effect of the intervention.

Table 4: ANCOVA analyses in which pre-test and post-test scores were controlled for ADSES and CTS

Source	Sum of Squares	df	Mean Square	F	p	η_p^2
Intervention Method						
ADSES	2,321	1	2,321	7,291	,008	,069
CTS	1,704	1	1,704	5,402	,022	,052

According to the Table 4 ANCOVA analysis results , when the students ADSES's pre-test post-test scores were analysed, it was seen that there was a statistically significant difference between the groups (F(1, 99)= 7,291, p<0.05; η_p^2 = 0.069). According to the adjusted mean scores, this difference was in favour of the control group ($\bar{X}$Google_Blockly= 3,72; $\bar{X}$Toxicode= 3,42). When the CTS pre-test post-test scores were examined, it was seen that there was also a statistically significant difference between the groups (F(1, 99)= 5,402, p<0.05; η_p^2 = 0.052). This difference was in favour of the control group according to the adjusted mean scores ($\bar{X}$Google_Blockly= 3,74; $\bar{X}$Toxicode= 3,48).

Finally, an independent sample t-test (Test F) was employed to examine the difference between the pre-test and post-test scores of the Experiment-1 and Control-1 groups. The results of this test are presented in Table 5.

Table 5: Independent Sample t-Test Analysis for Experiment-1 and Control-1 Groups

Scales/Tests	Groups	Classes	n	$\overline{X}$	sd	t	df	p
ADSES	Experimental-1	5-D	26	0.45	0.35	-2.14	51	0.03
	Control-1	5-F	27	0.69	0.46			
CTS	Experimental-1	5-D	26	0.39	0.25	-1.56	51	0.12
	Control-1	5-F	27	0.54	0.45			

As seen in Table 5, there is no statistically significant difference between the experimental-1 and control-1 groups in terms of CTS, although the mean score difference of the control-1 group is higher (t(51)=-1,56; p>0.05). On the other hand, there is a statistically significant difference between the control-1 and experiment-1 groups in ADSES mean scores in favour of the control-1 group (t(51)=-2,14; p<0.05).

4 Discussion and Conclusion

In this study, which was designed with Solomon four-group experimental model to investigate the effect of Toxicode and Google Blockly applications used in block-based coding education on 5th grade students‘ algorithm development self-efficacy perceptions and computational thinking skills, it was concluded that while the averages of students' algorithm development self-efficacy perceptions and computational thinking skills increased in both applications, this increase was statistically significant only in the algorithm development self-efficacy perception in favour of the control group using Google Blockly application.

In general, Toxicode and Google Blockly applications significantly increased students' computational thinking skills. Similarly, Totan (2021) concluded that students' self-efficacy perceptions towards computational thinking skills increased after block-based coding training with Google Blockly. Similarly, Ceylan (2020), in his study on the effect of Scratch teaching based on scenario application on computational thinking skills and problem-solving skills, showed that this education had a positive effect on students' academic achievement and retention scores. On the other hand, Atiker (2019) observed that coding instruction through Scratch application had a positive effect on computational thinking skills. In another research, Ünsal (2020) stated that block-based coding applications using code.org, another similar platform, had a positive effect on the computational thinking skills and activity perception of 2nd grade primary school students. Ulucan (2022) also found that coding and robotics activities applied in his study with preschool students had a significant effect on computational thinking and problem-solving skills. Oluk et al. (2018) and Yünkül et al. (2017) also obtained similar results in their studies. These results are similar to the results of the current study. Also, Genç & Karakuş (2011), Vatansever (2018), and Bakırcı (2019) concluded that teaching programming via Scratch platform increased students' algorithm development self-efficacy perceptions.

In addition to numerous supportive studies in the literature, oppositly Atman Uslu et al. (2018) concluded that Scratch application activities did not affect students' computational thinking skills. In a similar study, Alagöz (2022) concluded that taking a coding course before did not show a significant difference in CT skills compared to not taking a coding course. In addition to these, it can be said that block-based coding applications facilitate abstract thinking, reduce errors, improve algorithm and logic skills, encourage creativity, support cooperation and sharing, and make more people interested in programming by providing a low threshold for starting ((Pinto & Escuderio, 2014; Resnick et al., 2009)). It has been reflected in the literature studies (Başaran, 2018; Secer, 2020; Ke, 2014; Ünal, 2019) that using block-based programming tools can positively increase students' attitudes towards courses and course success in different discipline areas.

Although this study was designed using a quantitative model, based on the observations of the information technologies and software course teacher during the intervention process, it can be stated that the interface of the Toxicode application is simple, understandable, easy and enjoyable, and that it is suitable for repetition, receiving feedback and suitable for students' own learning speed. The most appropriate and effective platforms for programming education can be investigated with qualitative and mixed-design studies that focus on the participant experience and examine their views on programming environments.

References

Akçay, A., Çoklar, A.N. (2016). Bilişsel Becerilerin Gelişimine Yönelik Bir Öneri: Programlama Egitimi. In A. İşman, H. F. Odabaşı ve B. Akkoyunlu (Eds.), *Egitim Teknolojileri Okumaları 2016*, (p.p. 131-139). Ankara: The Turkish Online Journal of Educational Technology (TOJET).

Alagöz, S. (2022). *Ortaokul ögrencilerinin bilgi işlemsel düşünmesine etki eden faktörler.* (Pub-

lication No. 792544) Master's thesis, Bolu Abant İzzet Baysal University. Turkish Council of Higher Education Thesis Center.

Alp, Y. (2019). *Blok tabanlı programlama ögretiminin ortaokul ögrencilerinin problem çözme becerisine ve bilgisayara yönelik tutumuna etkisi.* (Publication No. 556354) Master's thesis, İnönü University. Turkish Council of Higher Education Thesis Center.

Atman Uslu, N., Mumcu, F & Eğin, F. (2018). Görsel programlama etkinliklerinin ortaokul ögrencilerinin bilgi-işlemsel düşünme becerilerine etkisi. *Ege Egitim Teknolojileri Dergisi, 2*(1), 19-31.

Atiker, B. (2019). *Programlama ögretiminde ortaokul ögrencilerinin bilgi işlemsel düşünme becerilerinin başarıya etkileri.* (Publication No. 561543) Doctoral dissertation, İstanbul University. Turkish Council of Higher Education Thesis Center

Aytekin, A., Sönmez-Çakır, F., Yücel, Y.B. & Kulaözü, İ. (2018). Algoritmaların hayatımızdaki yeri ve önemi. *Avrasya Sosyal ve Ekonomi Araştırmaları Dergisi(Asead)*, *5*(7), 151-162.

Bakırcı, F. (2019). *Blok tabanlı programlama aracının 6. Sınıf ögrencilerinin programlama başarısı, algoritma geliştirme öz yeterlikleri ve güdülenmelerine etkisi.* (Publication No. 585158) Master's thesis, Sakarya University. Turkish Council of Higher Education Thesis Center.

Barr, D., Harrison, J. & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, *38*(6), 20-23.

Başaran, B. (2018). *Arduino'nun elektrik deneylerine entegre edilmesinin ve deney raporlarının poster şeklinde hazırlanmasının, fen bilgisi öğğretmen adaylarının fizik laboratuvarlarına, teknolojiye ve bilgi ve iletişim teknolojilerine yönelik tutumlarına etkisinin incelenmesi.* (Publication No. 494788) Master's thesis, Kocaeli University. Turkish Council of Higher Education Thesis Center.

Batni, B., Junaini, S. N., Sidi, J., Mustafa, W. A & Ismail, Z.I.A (2025). Current research trends of scratch block based programming for K-12: A systematic review. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, *51*(2), 138-152.

BattelleforKids (2023). *21st century learning experience*, `https://www.battelleforkids.org/networks/p21/frameworks-resources`

Braver, M.W., Braver, S.L. (1988). Statistical treatment of the Solomon four-group design: A meta-analytic approach *Psychological Bulletin*, *104*(1), 150-154.

Brennan, K., Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In F. J. Levine (Ed.), *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*, (pp. 1-25). The University of British Columbia.

Büyüköztürk, Ş. (2007). *Sosyal Bilimler için Veri Analizi El Kitabı.* Ankara: Pegem A Yayıncılık.

Campbell, D.T., Stanley, J.C. (1963). *Experimental and quasi-experimental designs for research.* Chicago: Rand McNally and Company.

Ceylan, V.K. (2020). *Senaryo temelli Scratch öğretim programının öğrencilerin bilgi işlemsel düşünme becerilerine, problem çözme ve programlama ünitesi erişilerine etkisi.* (Publication No. 603629) [Doctoral dissertation, Aydın Adnan Menderes University]. Turkish Council of Higher Education Thesis Center.

Cohen, L., Manion, L. & Morrison, K. (2007). *Research methods in education.* Routledge.

Çağıltay, N.E., Fal, M. (2014). *Scratch ile Programlama Ögreniyorum.* Ankara: ODTÜ Yayıncılık.

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*, 90-104.

Deniz, G. (2021). *Programlama egitiminde Tinkercad kullanımının ögrencilerin bilgi işlemsel düşünme becerisine ve algılarına etkisi.* (Publication No. 628725) [Master's thesis, Gazi University]. Turkish Council of Higher Education Thesis Center.

Denning, P.J. (2003). Great principles of computing. *Communications of the ACM*, *46*(11), 15-20.

Dönmez Usta, N., Turan Güntepe, E. (2019). Bilişim teknolojileri rehber ögretmenlerinin programlama araçlarına ilişkin deneyimlerinin incelenmesi *Amasya Üniversitesi Egitim Fakültesi Dergisi*, *8*(2), 373-396.

Duncan, D.I., Bell, T. & Tanimoto, S.L. (2014). What's the big idea? Toward a theory of programming education *Communications of the ACM*, *57*(10), 44-52.

Fessakis, G., Komis, V., Dimitracopoulou, A. & Prantsoudi, S. (2019). Overview of the computer programming learning environments for primary education. *Review of Science, Mathematics and ICT Education*, *13*(1), 7-33.

Fraser, N. (2015, October). Ten things we've learned from Blockly. In F. Turbak, D. Bau, J. Gray, C. Kelleher and J. Sheldon (Eds.)*Proceedings of the 2015 IEEE Blocks and Beyond Workshop*, 49-50.

Fraenkel, J.R., Wallen, N.E. (2008). *How to design and evaluate research in education. (8th ed.).* New York: McGraw-Hill.

Futscheck, G. (2006). Algorithmic thinking: The key for understanding computer science [Paper presentation]. In *2nd International Conference on Informatics in Secondary Schools, Vilnius, Lithuania*, `https://link.springer.com/chapter/10.1007/11915355_15`

Genç, Z., Karakuş, S. (2011, September). Tasarımla ögrenme: Egitsel bilgisayar oyunları tasarımında Scratch kullanımı, In *5th International Computer & Instructional Technologies Symposium* , 22-24.

Guadalupe, R., Fumarola, F. (2015). Teaching coding in schools: A new era for computer science education *Informatics In Education*, *14*(1), 1-16.

Grover, S., Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38-43.

Henderson, P. (2008). Computer science unplugged. *Journal of Computing Sciences in Colleges*, *23*(3), 168.

Kalelioglu, 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.

Karasar, N. (2012). *Bilimsel Araştırma Yöntemi (23rd ed.).* Ankara: Nobel Yayıncılık.

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *JComputers and Education*, *73*, 26-39.

Kert, S.B., Ugraş, T. (2009, May). Programlama eğitiminde sadelik ve eglence: Scratch örnegi. In *The First International Congress of Educational Research*, Çanakkale, Türkiye.

Kirk, R.E. (2009). Experimental design. In Millsap, R. and Maydeu-Olivares, A. (Eds.), *Sage Handbook Of Quantitative Methods in Psychology*. Thousand Oaks, CA: Sage

Korkmaz, Ö., Çakır, R. & Özden, M.Y. (2015). Bilgisayarca düşünme beceri düzeyleri ölçeginin (BDBD) ortaokul düzeyine uyarlanması. *Gazi Egitim Bilimleri Dergisi*, *1*(2), 67-86.

Korkmaz, Ö., Çakır, R. & Özden, M.Y. (2017). A validity and reliability study of the Computational Thinking Scales (CTS). *Computers in Human Behavior*, *72*, 558-569.

Korkmaz, Ö., Balcı, H., Çakır, R. & Uğur Erdogmus, F. (2021). Görsel programlama ortamlarında yapılan oyun geliştirme etkinliklerinin etkililigi. *Mehmet Akif Ersoy Üniversitesi Egitim Fakültesi Dergisi*, *57*, 52-73.

Kraleva, R., Kralev, V., & Kostadinova, D. (2019). A Methodology for the analysis of block-Based programming languages appropriate for children. *Journal of Computing Science and Engineering JCSE*, *13*(1), 1-10.

MoNE[MEB]. (2018, October 22). *Bilişim teknolojileri ve yazılım dersi ögretim programı (Ortaokul 5 ve 6.Sınıflar)*. `http://mufredat.meb.gov.tr/ProgramDetay.aspx?PID=374`

Morelli, R., De Lanerolle, T., Lake, P., Limardo, N., Tamotsu, E. & Uche, C. (2011). Can Android app inventor bring computational thinking to K-12. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, (pp. 1-61), Association for Computing Machinery.

Oluk, A., Korkmaz, Ö. (2016). Comparing students' scratch skills with their computational thinking skills in terms of different variables. *I. J. Modern Education and Computer Science*, *8*(11), 1-7.

Oluk, A., Korkmaz, Ö. & Oluk, H.A. (2018). Scratch'ın 5. Sınıf ögrencilerinin algoritma geliştirme ve bilgi işlemsel düşünme becerilerine etkisi. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *9*(1), 54-71.

Özdamar, K. (2013). *Paket programları ile istatistiksel veri analizi*. Ankara: Pelikan Yayıncılık.

Özel, O. (2019). *Programlama yöntemlerinin ortaokul ögrencilerinin bilgi işlemsel düşünme becerisine yönelik öz yeterlik algısına ve programlama başarısına etkisi*. (Publication No. 602802) [Master's thesis, Marmara University]. Turkish Council of Higher Education Thesis Center.

Papert, S. (1980). *Children, computers and powerful ideas*. New York: Basic Books.

Pinto, A., Escuderio, P. (2014, June). The use of Scratch for the development of 21st century learning skills in ICT *In 2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, 1-4.

Rovshenov, A., Sarsar, F. (2023). Research trends in programming education: a systematic review of the articles published between 2012-2020 *Journal of Educational Technology & Online Learning*, *6*(1), 48-81.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K. & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67.

Sonjaya, R.P., Munir, M. (2025). Examining the impact of block-based visual programming in programming education: A systematic review. *Indonesian Journal of Teaching in Science*, *5*(1), 11-20.

Sayın, Z., Seferoglu, S. (2016). yeni bir 21. Yüzyıl becerisi olarak kodlama egitimi ve kodlamanın egitim politikalarına etkisi [Paper presentation]. In *XVIII. Akademik Bilişim Konferansı*, Aydın, Türkiye.

Secer, M. (2020). *Bilişim teknolojileri ve yazılım dersinde arduino kodlama ile kagıt-kalem kodlama uygulamalarının ögrencilerin bilgi işlemsel düşünme becerileri, problem çözme becerileri ve STEM tutumları üzerine etkisi.* (Publication No. 640814) Master's thesis, Mersin University. Turkish Council of Higher Education Thesis Center.

Sedgewick, R., Wayne, K. (2011). *Algorithms*. Princeton University: Pearson Education.

Shelby, C.C. (2014). *How can the teaching of programming be used to enhance computational thinking skills?*. Doctoral dissertation, University of Southampton.

Tabachnick, B.G., Fidell, L.S. (2013). *Using Multivariate Statistics (sixth ed.)*. Pearson.

Totan, H.N. (2021). *Blok tabanlı kodlama egitiminin ortaokul ögrencilerinin bilgi işlemsel düşünme becerileri ve kodlama ögrenimine yönelik tutumlarına etkisi: Blocky örnegi.* (Publication No. 28840638) Master's thesis, Necmettin Erbakan University. ProQuest Dissertations & Theses.

Ulucan, J. (2022). *Kodlama ve robotik etkinliklerinin bilgi-işlemsel düşünme ve problem çözme becerilerine etkisi: Meta-analiz çalışması.* (Publication No. 753059) Master's thesis, Yıldız Teknik University. Turkish Council of Higher Education Thesis Center.

Ünal, A. (2019). *Sosyal bilimler lisesi ögrencilerine blok tabanlı programlama ögretiminin kaygı, bilişsel yük ve başarıya etkisi.* (Publication No. 551146) Master's thesis, Atatürk University. Turkish Council of Higher Education Thesis Center.

Ünsal, İ. (2020). *Blok tabanlı programlama etkinliklerinin ilkokul 2. sınıf ögrencilerinin bilgi işlemsel düşünme becerisi ve etkinlik algısı üzerine etkisi.* (Publication No. 643498) Master's thesis, Sakarya University. Turkish Council of Higher Education Thesis Center.

Wing, J.M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* , *366*(1881), 3717-3725.

Vatansever, Ö. (2018). *Scratch ile programlama ögretiminin ortaokul 5. ve 6. sınıf ögrencilerinin problem çözme becerileri üzerindeki etkisinin incelenmesi* . (Publication No. 501053) Master's thesis, Uludag University. Turkish Council of Higher Education Thesis Center.

Yıldırım, A., Şimşek, H. (2011). *Sosyal bilimlerde nitel araştırma yöntemleri* . Ankara: Seçkin Yayıncılık.

Yüksel, S. (2017). *Scratch programı ögretiminde ayrılıp birleşme teknigi kullanımının ögrencilerin derse yönelik tutumuna akademik başarısına ve kalıcılıga etkisi* . (Publication No. 472231) Master's thesis, Adnan Menderes University. Turkish Council of Higher Education Thesis Center.

Yükseltürk E., Altıok, S. (2015). Bilişim teknolojileri ögretmen adaylarının bilgisayar programlama ögretimine yönelik görüşleri. *Amasya Üniversitesi Egitim Fakültesi Dergisi*, *4*(1), 50-65.

Yünkül, E., Durak, G., Çankaya, S. & Abidin, Z. (2017). The effects of scratch software on students' computational thinking skills. *Necatibey Egitim Fakültesi Elektronik Fen ve Matematik Egitimi Dergisi (EFMED)*, *11*(2), 502-517.