Id: 1907166

Participatory Educational Research (PER)
Vol.13(6), pp. 101-123, November 2026
Available online at http://www.perjournal.com
ISSN: 2148-6123
http://dx.doi.org/10.17275/per.26.80.13.6

The Effect of Coding Education on Gifted Students' Problem Solving Skills, Attitude Towards Coding, and Self-Efficacy

Sinem Görgü ÇIRPAN*
Aydın Adnan Menderes University, Institute of Social Sciences, Master's Degree, Aydın, Türkiye
ORCID: 0009-0006-7431-4987

Serdar ÇİFTCİ
Faculty of Education Computer Education and Instructional Technology, Aydın Adnan Menderes University, Türkiye
ORCID: 0000-0001-5282-1861

Article history

Received:
10.03.2026

Received in revised form:
13.05.2026

Accepted:
17.06.2026

Key words:
scratch, code.org, coding education, problem solving skills, self-efficacy, attitude

The purpose of this study is to examine the effect of coding education provided to 5th grade students studying at the Science and Art Center (BİLSEM) on students' attitudes towards coding, problem-solving skills, self-efficacy levels, and students' opinions on Scratch and Code.org software. In the study, coding education was provided to 30 fifth-grade students at BİLSEM in Nazilli district of Aydın, Türkiye, during the 2023-2024 academic year. The study included two experimental groups. One group was trained using Scratch, the other using Code.org. As a result of the research, it was concluded that there was no significant difference in the problem-solving skills of the students who received coding training with Scratch, but there was a statistically significant difference in the self-control dimension of the problem-solving skills of the students who received coding training with Code.org. It was concluded that there was no significant difference in the attitudes of students who received coding training with Scratch, while there was a significant difference in the attitudes of students who received coding training with Code.org. There was a significant increase in the complex programming skills of students who received Code.org training and in the skills of simple programming tasks of students who received Scratch training after the training. Gifted students stated that coding with the block-based programming tool was easy, fun, and enjoyable after the training.

Introduction

The continuous development of technology is also driving innovation in education. Instead of teacher-centered education, the traditional educational approach used in schools, learner-centered education should be implemented using technological advancements (Akpınar & Gezer, 2010). In the current situation, it is crucial for learners to acquire 21st-century skills (Karataş, 2021). The most fundamental skills that individuals need to possess in the 21st century are defined as problem solving, communication, collaboration, critical

* Correspondency: sciftci@gmail.com

thinking, information and technology literacy, flexibility and adaptability, global competencies, and financial literacy (Geisinger, 2016). Scott (2015) stated that educational programs must change and new learning styles are needed in courses for individuals to acquire these skills. Programming education is a tool that enables the development of 21st-century skills (Sayın & Seferoğlu, 2016). By facilitating complex project design and logical reasoning, programming environments allow students to navigate multifaceted challenges effectively (Bozan & Taşlıdere, 2024). Furthermore, the utilization of block-based visual environments is increasingly recognized for its capacity to foster high-level cognitive functions, including analytical and computational thinking (Sungur & Ozkan, 2026). Moreover, systematic coding training significantly improves metacognitive awareness among gifted learners by allowing them to monitor their problem-solving steps and iteratively refine their logic (Yurtbakan, 2022).

Programming is the process of enabling hardware and software to communicate, giving commands to the computer to solve an encountered problem, and developing applications using programming languages (Saygıner & Tüzün, 2017; Sayın & Seferoğlu, 2016). Students' creativity, reasoning, and problem-solving skills improve with programming education (Calder, 2010; Kobsiripat, 2015). There are also studies indicating that programming education is complex and difficult. Özmen and Altun (2014) stated that students fail in programming due to lack of practice, inability to create algorithms, and insufficient knowledge. Saygıner and Tüzün (2017) emphasized that the reasons for the difficulties experienced in programming education are due to not fully learning programming logic, being unfamiliar with the programming language, and teaching methods. Karaduman and Akpınar (2021) stated that difficulties are experienced due to a lack of repetition and practice, rote-based education, and a lack of abstract thinking. To overcome these difficulties experienced in programming education, it is recommended that individuals new to programming choose block-based programming tools (Balcı, 2024; Saygıner & Tüzün, 2017). These platforms facilitate learning by utilizing visual, drag-and-drop interfaces that reduce the cognitive load associated with complex syntax, thereby allowing students to focus on fundamental computational thinking and algorithmic design (Tsai et al., 2025).

Programming can be discussed under two main headings: text-based programming and block-based programming. Text-based programming is a programming structure built using words, code, and specific syntax rules. Block-based programming is programming created using pre-built code blocks using a visual interface and drag-and-drop. Block-based programming is software created by combining nested code blocks using the drag-and-drop method (Haymana & Özalp, 2020). Block-based programming is preferred over text-based programming for individuals new to programming. Çatlak, Tekdal, and Baz (2015) emphasized that when creating projects with block-based programming languages, users create projects using code blocks in a fun and easy way, helping them learn abstract concepts that are difficult to learn. This visual approach mitigates the discouragement often associated with traditional syntax memorization, thereby increasing student motivation and persistence in mastering complex computational tasks (Erümit & Şahin, 2025).

Individuals are exposed to technology at an early age. Visual programming tools are used to develop the programming skills of these individuals who are exposed to technology at an early age. These individuals, whose programming skills develop at an early age, will be encouraged to generate new ideas and develop projects (Yadav et al., 2016). Individuals who are introduced to programming early are observed to be able to use these skills at a higher level and are more successful in the field of informatics. Students who are introduced to

Code.org at an early age develop computational thinking skills and the ability to solve real-life problems (Barradas et al., 2020). A study by Fessakis, Gouli, and Mavroudi (2013) found that acquiring coding skills at a young age in 5- and 6-year-old children improved their problem-solving skills and higher-order, algorithmic thinking.

Today, the internet is present in every aspect of our lives. The continuous development of the internet has enabled the introduction of the concepts of the internet of things and artificial intelligence. With the internet of things, devices can connect and communicate with each other over the internet. This suggests that the need for individuals with programming skills will constantly increase (Karataş, 2021). It is believed that individuals who acquire these skills will facilitate interdisciplinary knowledge exchange. Learning programming prepares individuals for the future by designing algorithms, solving problems, and acquiring productive skills (Karataş, 2021). Complex syntax and incomprehensible interfaces make it difficult for students to learn programming languages. Using tools with visual features, understandable interfaces, and syntax facilitates learning programming languages. Yükseltürk and Altıok (2015) state that programming is difficult and complex, and therefore, beginners are developing their programming skills using visual programming languages such as Scratch, Small Basic, Alice, Lego Mindstorm, Etoys, and Hyperstudio to minimize the complexity of programming.

Students can easily use block-based programming tools starting from preschool. Saygıner and Tüzün (2017) stated that block-based programming tools can be used by any student from preschool to high school, helping them become productive individuals. Özçelik and Akgündüz (2018) stated that gifted students, whose needs and personal characteristics differ from typically developing students, should be introduced to programming in a planned manner from an early age. Moreover, integrating these tools into the curriculum supports the acquisition of essential 21st-century competencies, such as media literacy and information technology skills, which are critical for navigating a globalized information society (Kalelioğlu & Gülbahar, 2014). Furthermore, the application of these environments fosters a dynamic mindset, enabling students to address authentic problems through collaborative and creative inquiry (Sungur & Ozkan, 2026).

Gifted students are children who outperform or achieve high levels of success in a specific field compared to their peers. Gifted individuals have advanced cognitive capacities compared to their peers and require differentiated instruction to enhance their cognitive capacity (Bakioğlu & Levent, 2013). STEM (Science, Technology, Engineering, Mathematics) education designed for gifted students is thought to improve students' attitudes toward robotics and coding (Avcı, 2021). A study by Bozan (2022) concluded that coding education provided to gifted students increases student achievement and self-efficacy. In the study conducted by Toklu (2019), it was stated that gifted students learned programming logic by using the Kodu Game Lab program and that gifted students with high levels of curiosity and interest were encouraged to learn game design and coding education.

It is believed that the development of our society requires gifted individuals to receive various training programs to ensure their development in every aspect (Kurtdaş, 2023; Oruç & Çağır, 2020). Avcı (2021) stated that supporting the education of gifted individuals with STEM and coding training will help them acquire 21st-century skills. Consequently, the importance of coding training for gifted individuals to discover their potential is increasing daily, and studies on this topic are needed. There are many studies on coding education in various disciplines such as students' problem-solving skills, attitudes towards coding, academic achievement,

computational thinking, and self-efficacy (Bala, 2019; Dinçer, 2018; Emiroğlu, 2021; Gürbüztürk & Tanataş, 2024; Karaduman, 2019; Kırçalı, 2019; Lahullier, 2019; Okuducu, 2020; Otu, 2020; Papadakis et al., 2016; Zdawczyk & Varma, 2023). In these studies, the importance of coding education is emphasized. Dinçer (2018) concluded that students' self-efficacy perceptions towards programming increased with Scratch and Kodu Game Lab. In the study conducted by Karaduman (2019), it was concluded that students' attitudes were positively affected and their learning levels increased with Code.org. Totan (2021) conducted a study on a block-based program and observed that computational thinking skills were positively affected. A study by Taş (2018) found significant differences in the sub-dimensions of computational thinking, algorithmic thinking, and collaborative and creative work in students receiving computer-assisted mathematics instruction. Calder (2010) reported that students using Scratch improved their problem-solving skills and found the Scratch language engaging.

The main objective of this study is to evaluate the influence of a 10-week block-based programming intervention—incorporating both Code.org and Scratch platforms—on the problem-solving competencies, attitudes toward coding, and programming self-efficacy of gifted fifth-grade students, while simultaneously examining their lived experiences throughout the intervention. To address this overarching aim, a mixed-methods research design was implemented, structured around the following specific quantitative and qualitative research objectives:

(1) To determine whether the 10-week block-based programming intervention significantly improves the problem-solving skills of gifted fifth-grade students.
(2) To evaluate whether the intervention significantly influences students' attitudes toward educational computer game-supported coding.
(3) To assess whether the intervention significantly enhances students' self-efficacy perceptions regarding both basic and complex programming tasks.
(4) To explore the experiences, perceived challenges, and qualitative insights of gifted fifth-grade students regarding the block-based coding process throughout the 10-week implementation.

Research Method

This section covers the research model, sample, data collection tools, the data collection process, and data analysis.

Research Design

This study aimed to examine the effects of coding training on gifted students' attitudes toward coding, problem-solving skills, and self-efficacy levels. A mixed-method research method was employed, combining quantitative and qualitative research methods. An embedded design was employed as part of the mixed research method. An embedded design is a method in which one type of data (qualitative or quantitative) provides a supportive role for the other, allowing for a more complete understanding of the research problem (Fielding, 2012). The current study employed a quasi-experimental design, specifically a pretest-posttest design without a control group. Quasi-experimental designs are used in field settings where random assignment of participants to groups is not feasible or practical (Becker et al., 2017; Waddington et al., 2017). Qualitative data were collected using a semi-structured interview form to support and enrich the quantitative findings.

Sample

The study group consisted of gifted fifth-grade students enrolled at *BİLSEM* in the Nazilli district of Aydın, Türkiye, province during the 2023-2024 academic year. The study began with 32 students, but two students were excluded from the study due to insufficient participation and were not included in the analysis. Convenience sampling was used due to the easily accessible nature of the study group (Kılıç, 2013).

Table 1: Number of Study Group Members

Group Name	Number of Members
Scratch Experimental Group	15
Code.org Experimental Group	15
Total	30

When selecting the study's experimental groups, two different experimental groups were randomly assigned. One group received Scratch training, and the other received Code.org training. Information regarding the number of members in the experimental groups is provided below in Table 2.

Table 2: Some demographic information of the students in the treatment groups

Variable	Participant	F	%
Experimental Groups	Code.org	15	50.0
	Scratch	15	50.0
Gender	Male	14	46.7
	Female	16	53.3
Programming Time	1 years	14	46.7
	2 years	7	23.3
	3 years	7	23.3
	4 years	1	3.3
	5 years	1	3.3
	Total	30	100

The study consisted of 15 students in each treatment group. Fourteen of the participating students were male and sixteen were female. Because the intervention was conducted in the second semester, the students used a different programming tool in their first-semester information technologies and software course.

Data Collection Tools

Data collection tools included the problem-solving inventory for primary school children developed by Serin, Bulut, and Saygılı (2010); the attitude scale toward educational computer game-supported coding learning developed by Keçeci, Alan, and Zengin (2016); the self-efficacy perception scale for programming developed by Altun and Mazman (2012); and a semi-structured interview form.

Problem Solving Inventory for Primary School Children (PSIPC): Developed by Serin, Bulut, and Saygılı, this inventory measures the self-perceptions of primary and middle school students regarding their problem-solving capabilities. It comprises 24 items on a 5-point Likert-type scale, yielding scores across three sub-dimensions: Confidence in Problem-Solving Ability, Self-Control, and Avoidance. In the original validation study, the total internal consistency coefficient was reported as .80, with sub-dimension coefficients of .75, .78, and .66, respectively. For the current sample, the total Cronbach's alpha was .82,

confirming high internal reliability. Given the distinct cognitive profiles and metacognitive awareness of gifted students, this instrument was selected for its ability to operationalize abstract problem-solving concepts into behaviors suitable for fifth-grade concrete operational thinkers. Furthermore, the scale facilitates addressing research questions concerning whether block-based coding structures enhance self-regulation and confidence when students encounter logic errors.

Attitude Scale Toward Educational Computer Game-Supported Coding Learning (AS-ECG): Developed by Keçeci, Alan, and Zengin (2016), this scale was designed to evaluate students' emotional, cognitive, and social attitudes toward learning programming via gamified environments. The scale comprises 24 items evaluated on a 5-point Likert-type format, divided into three robust dimensions: Enthusiasm for Learning Coding, Game Usage, and Computer Effect on Socialization. The original construct validity was established through exploratory factor analysis (EFA), accounting for a high variance percentage, with an overall Cronbach's alpha of .86. In this study, the reliability coefficient for the entire scale was monitored and calculated as .85. Since both Code.org and Scratch leverage gamified progression and visual assets, assessing student interaction through a general computing scale would be insufficient. This tool was selected because it uniquely isolates game-supported coding attitudes, allowing the researchers to measure if the dynamic nature of platforms influences student learning desires over a 10-week period.

Self-Efficacy Perception Scale for Programming (SEPS-P): This scale was developed by Altun and Mazman (2012) to assess individuals' self-perceived competencies in executing programming tasks. It utilizes a 5-point Likert configuration containing two foundational factors: Simple Programming Tasks and Complex Programming Tasks. Construct validity was verified via structural equation modeling in its original adaptation, yielding an exceptionally high internal consistency of .93. For our targeted group at *BİLSEM*, the Cronbach's alpha was computed as 89. Measuring absolute achievement in coding often ignores the psychological barrier of programming anxiety. This scale was selected because block-based tools explicitly aim to reduce syntax complexity. Tracking changes in self-efficacy provides concrete evidence on whether visual block structures effectively transition students from simple scripts to complex nested loops.

Semi-Structured Interview Form (Qualitative Tool): The qualitative data collection tool was developed entirely by the current researchers to complement and contextualize the quantitative findings. The development followed a rigorous three-step protocol:

- To triangulate the quantitative data and provide deeper contextual insight, an original semi-structured interview instrument was developed by the researchers. The construction of this tool followed a rigorous three-phase methodological protocol:
- Drafting: An initial pool of 5 open-ended questions was generated based on the core pillars of the 10-week implementation plan (focusing on liked features, software limitations, and logic errors).
- Expert Panel Review: The draft was submitted to two field experts in Educational Technology and one specialist in Gifted Education to ensure content validity and age-appropriateness. Based on their feedback, phrasing was simplified to prevent leading responses.
- Pilot Testing: A pilot interview was conducted with two non-participant 5th-grade students to confirm the clarity and comprehensibility of the prompts.

To ensure qualitative rigor, post-intervention interviews were recorded digitally to mitigate the risk of data loss. Transcripts were subjected to systematic content and thematic analysis using MAXQDA Plus 2022. Inter-coder reliability was verified through an independent

review of transcripts and codebooks by a subject-matter expert alongside the primary researcher. Agreement was quantified using the formula developed by Miles and Huberman, yielding a consensus rate that met the requisite threshold for qualitative dependability. Recognizing that quantitative assessments identify the presence of change without elucidating the underlying mechanisms, semi-structured interviews were utilized to capture the authentic experiences of gifted students regarding specific challenges—such as block placement and interface navigation—as well as their intrinsic engagement with Scratch and Code.org.

Implementation Process

This study was conducted at the Nazilli Science and Art Center in the Nazilli district of Aydın, affiliated with the Ministry of National Education, during the second semester of the 2023-2024 academic year, 1 hour per week for 10 weeks. The study was conducted with 5th-grade students studying at the Nazilli Science and Art Center. The implementation plan of the study is given in Table 3.

Table 3: Implementation plan of the study

Weeks	Activities Performed
Week	Week 1: Pretest was administered.
Week	Week 2: Algorithm was introduced.
Week	Week 3: Application was introduced. Moving objects was introduced.
Week	Events was introduced.
Week	Loops was introduced.
Week	Conditions was introduced.
Week	Functions was introduced.
Week	Variables was introduced.
Week	Project was completed.
Week	Posttest was administered.

The implementation process was conducted over a 10-week period with one 60-minute session per week. The process was structured as follows: *(1) Preparation:* Initial administration of the pre-tests to establish a baseline for problem-solving skills, attitudes, and self-efficacy. *(2) Instructional Delivery:* Each session began with a 10-minute demonstration of the weekly concept (e.g., loops, conditional logic) on a shared screen. *(3) Active Engagement:* Students worked individually on the Scratch or Code.org platforms. The researcher provided guidance using a scaffolding approach, where students were encouraged to 'debug' their own logic rather than being given direct answers. *(4)Task Progression:* Activities moved from simple animations to more complex game-based scenarios, aligning with the 10-week plan outlined.

Rather than relying on a single platform, this study integrated both Code.org and Scratch to construct a scaffolded, pedagogical progression tailored to the developmental needs of gifted 5th-grade learners. Code.org served as the primary instructional tool during the initial phase, functioning as a structured, puzzle-based learning environment. This platform was selected for its diagnostic and introductory utility; its objective-driven progression enables students to master fundamental computational concepts through immediate algorithmic feedback, which mitigates syntax-related anxiety and initial frustration. Conversely, Scratch was introduced in the subsequent phase to facilitate the transition from computational consumption to open-ended computational creativity. By removing rigid structural constraints, Scratch provides a sandbox environment wherein students may apply previously acquired abstract logic to construct original, game-based scenarios. This dual-platform approach ensures that students establish technical self-efficacy through Code.org before transitioning to higher-order

divergent thinking and problem-solving through the open-ended design opportunities inherent in Scratch.

This study employs a one-group pretest-posttest pre-experimental research design. The exclusion of a traditional control group was necessitated by administrative and ethical considerations inherent to the specialized educational context. Specifically, at this center for gifted education, students are assigned to fixed, highly individualized project clusters; thus, denying a cohort participation in a 10-week advanced educational technology curriculum to serve as a passive control group was deemed ethically and logistically untenable. Furthermore, due to the significant cognitive heterogeneity inherent in gifted populations, identifying a matched control group with identical cognitive and motivational profiles is complex. Consequently, this study centers on intra-group variance, tracking the longitudinal development of the participants using their own pre-intervention data as benchmarks—a methodology supported by exploratory educational technology research.

To reinforce internal validity and minimize potential researcher bias, the intervention was operationalized into three distinct roles. First, as the *Curriculum Designer*, the researcher systematically structured the 10-week syllabus to ensure a linear progression in computational complexity across both platforms. Second, in the capacity of *Pedagogical Facilitator*, the researcher eschewed monologic lecturing in favor of a supportive approach; after a 10-minute introduction to the weekly algorithmic challenge, the researcher utilized scaffolded questioning to foster autonomous student debugging rather than providing direct solutions. Finally, as the *Data Collector*, the researcher ensured consistency by administering quantitative inventories under standardized environmental conditions and employing uniform prompts during semi-structured interviews to mitigate the risk of influencing participant responses.

Data Analysis

Quantitative Data Analysis

The normality distributions and descriptive statistics of the scale scores obtained by students in the Code.org and Scratch applications were analyzed using SPSS v25, and separate skewness and kurtosis coefficients were calculated for both the Code.org and Scratch groups. The normality distribution of the data was examined by considering the skewness and kurtosis coefficients. George and Mallery (2016) stated that if the skewness and kurtosis coefficients are within the range of +-2, the data are normally distributed.

Although the Central Limit Theorem suggests that parametric tests can be robust even with non-normal distributions in larger samples (Kwak & Kim, 2017), the decision to utilize non-parametric tests in this study was theoretically justified by the nature of the data. First, the variables were measured using Likert-type scales, which yield ordinal data. In such cases, rank-based non-parametric tests are often more appropriate as they do not assume an interval relationship between scale points (Sung & Wu, 2018). Furthermore, literature suggests that even when CLT-related thresholds are considered, formal judgments of normality are essential, and non-parametric methods remain a conservative and reliable alternative when specific assumptions (such as kurtosis limits) are not fully met (Bishara et al., 2021; Koh & Ahad, 2020).

Table 4: Descriptive statistics of the pre-test and post-test scores of students in the Code.org group

Group	Scale	Variable	N	$\bar{X}$	S.d.	Skewness	Kurtosis
Pre-test t	Perception of Self-Efficacy for Programming	Simple Programming Task	15	13.73	5.16	.094	-1.192
		Complex Programming	15	20.00	5.52	1.022	1.102
	Children's Problem-Solving Inventory	Confidence in Problem-Solving Ability	15	43.40	6.84	-.254	.695
		Self-Control	15	25.27	6.09	.003	-.705
		Avoidance	15	19.33	4.13	-1.715	1.397
	Attitude Scale Towards Educational Computer Game-Supported Coding Learning	Enthusiasm for Learning Coding	15	23.67	6.00	.117	-1.300
		Game Usage	15	29.80	5.85	-.838	1.025
		Computer Effect on Socialization	15	13.27	2.91	.195	-1.329
Post-Test	Perception of Self-Efficacy Regarding Programming	Simple Programming Task	15	15.20	5.24	-.423	-1.368
		Complex Programming	15	24.53	7.47	-.772	-.422
	Children's Problem-Solving Inventory	Confidence in Problem-Solving Skills	15	44.47	8.61	-.619	-.476
		Self-Control	15	28.27	2.96	-.642	-.349
		Avoidance	15	20.07	3.31	-1.793	1.324
	Attitude Towards Educational Computer Game-Supported Coding Learning Scale	Willingness to Learn Coding	15	27.87	8.61	1.174	1.960
		Game Use	15	33.33	3.74	.248	-.883
		Computer Effect on Socialization	15	12.93	2.52	-.747	.838

An examination of the pretest and posttest scores of students in the Code.org group reveals that the skewness and kurtosis coefficients are within the +-2 range. Accordingly, it can be said that the data are within the normal distribution range (George & Mallery, 2016).

Table 5: Descriptive statistics of the pretest and posttest scores of students in the Scratch group

Group	Scale	Variables	N	$\bar{X}$	S.d.	Skewness	Kurtosis
Pretest	Perception of Self-Efficacy for Programming	Simple Programming Task	15	12.93	4.86	.068	.106
		Complex Programming	15	18.20	9.93	.689	-1.057
	Children's Problem-Solving Inventory	Confidence in Problem-Solving Ability	15	46.93	8.36	-.893	-.268
		Self-Control	15	27.00	5.33	-1.670	2.081
		Avoidance	15	20.87	2.90	-.679	.983
	Attitude Scale Towards Educational Computer Game-Supported Coding Learning	Enthusiasm for Learning Coding	15	22.33	7.72	.533	-.485
		Game Usage	15	30.07	8.09	.051	-.847
		Computer Effect on Socialization	15	14.33	2.79	-.102	-.588
Post-Test	Perception of Self-Efficacy Regarding Programming	Simple Programming Task	15	15.80	2.73	.728	.088
		Complex Programming	15	21.47	8.53	.684	1.482
	Children's Problem-Solving Inventory	Confidence in Problem-Solving Skills	15	47.60	7.01	-.808	1.152
		Self-Control	15	28.73	4.88	-.683	-.597

	Avoidance	15	21.47	2.70	-.374	-.586
Attitude Towards Educational Computer Game-Supported Coding Learning Scale	Willingness to Learn Coding	15	23.60	8.29	.770	-.017
	Game Use	15	32.73	7.01	-.292	-1.034
	Computer Effect on Socialization	15	13.73	3.03	.116	-1.161

An examination of the pretest and posttest scores of the students in the Scratch group revealed that the skewness and kurtosis coefficients were within the +-2 range. Accordingly, the pretest and posttest data in both groups met the conditions of normality (George & Mallery, 2001; Newell & Hancock, 1984). Accordingly, an independent samples t-test was used to compare the Code.org and Scratch groups, and a dependent samples t-test was used to compare the pretest and posttest scores.

Qualitative Data Analysis

Students' responses to the semi-structured interview form were recorded with a voice recorder. This prevented incomplete data from the interviews. After the training, the students created themes based on the coding platforms they used, Code.org and Scratch. A concept map was created using the MAXQDA Plus 2022 software package based on the students' opinions, and the responses were categorized and organized into themes. Thematic analysis, especially when working with complex and rich qualitative data, facilitates an in-depth understanding of the data and helps develop comprehensive answers to research questions. Furthermore, Braun and Clarke (2006) define thematic analysis as the process of carefully coding data, grouping these codes into themes based on their common characteristics, and finally structuring these themes into the main findings of the research. The data obtained from the interviews were reviewed by a field expert, and separate codebooks were created. The interviews were then compiled into a single codebook based on the consensus of the field expert and the researcher, and content analysis was conducted to ensure the reliability of the data obtained.

Research Findings

Quantitative Findings

Pre-test comparisons of students' perceptions of self-efficacy in programming in the Code.org and Scratch groups were examined using an Independent Samples t-test and are presented in Table 6.

Table 6: Pre-test comparison of students' perceptions of self-efficacy in programming

Variable	Group	N	$\overline{X}$	S.s	sd	t	p
Simple Programming	Code.org	15	13.73	5.16	28	.437	.665
	Scratch	15	12.93	4.86			
Complex Programming Task	Code.org	15	20.00	5.52	28	.614	.544
	Scratch	15	18.20	9.93			

$p<0.05$

When the students' pretest scores on the Programming Self-Efficacy Perceptions before the Code.org and Scratch training were compared, it was concluded that there was no statistically significant difference ($p>0.05$). Considering this, it can be said that the students displayed a homogeneous distribution before the training. The pretest comparisons of the Children's

Problem Solving Inventory for students in the Code.org and Scratch groups were examined using an Independent Samples t-test and are presented in Table 7.

Table 7: Students' pretest comparisons of the Children's Problem Solving Inventory

Variable	Group	N	$\overline{X}$	S.s	sd	t	p
Confidence in Problem-Solving Skills	Code.org	15	43.40	6.84	28	-1.266	.216
	Scratch	15	46.93	8.36			
Self-Regulation	Code.org	15	25.27	6.09	28	-.830	.414
	Scratch	15	27.00	5.33			
Avoidance	Code.org	15	19.33	4.13	28	-1.176	.250
	Scratch	15	20.87	2.90			

p<0.05

When the students' pretest scores on the Children's Problem Solving Inventory before the Code.org and Scratch training were compared, no statistically significant difference was found (p>0.05). Considering this, it can be said that the students displayed a homogeneous distribution before the training.

Pretest comparisons of the Attitude Scale towards Educational Computer Game-Supported Coding Learning of students in the Code.org and Scratch groups were examined using an independent t-test and are presented in Table 8.

Table 8: Pretest comparisons of the Attitude Scale towards Educational Computer Game-Supported Coding Learning of students

Variable	Group	N	$\overline{X}$	S.s	sd	t	p
Willingness to Learn Coding	Code.org	15	23.67	6.00	28	.528	.601
	Scratch	15	22.33	7.72			
Game Use	Code.org	15	29.80	5.85	28	-.103	.918
	Scratch	15	30.07	8.09			
Computer Impact on Socialization	Code.org	15	13.27	2.91	28	-1.023	.315
	Scratch	15	14.33	2.79			

p<0.05

When the students' pretest scores on the Attitude Scale Towards Educational Computer Game-Supported Coding Learning were compared before the Code.org and Scratch training, no statistically significant difference was found (p>0.05). Considering this, it can be said that the students exhibited a homogeneous distribution before the training.

The pretest-posttest comparisons of the Programming Self-Efficacy Perceptions of the students in the Code.org and Scratch groups were examined using a Dependent Samples t-test and are presented in Table 9.

Table 9: Pretest-posttest comparison of students' programming self-efficacy perceptions

Variable	Group	Test	N	$\overline{X}$	S.s	sd	t	p
Code.org	Simple Programming Task	Pretest	15	13.73	5.16	14	-.993	.337
		Posttest	15	15.20	5.24			
	Complex Programming	Pretest	15	20.00	5.52	14	-2.183	.047*
		Posttest	15	24.53	7.47			
Scratch	Scratch Simple Programming Task	Pretest	15	12.93	4.86	14	-2.484	.026*
		Posttest	15	15.80	2.73			
	Complex Programming	Pretest	15	18.20	9.93	14	-1.714	.109
		Posttest	15	21.47	8.53			

p<0.05

Students' pretest and posttest scores on their Perception of Self-Efficacy in Programming before and after the Code.org and Scratch training were compared. Accordingly, a statistically significant difference was found in the Complex Programming dimension in the Code.org training ($p<0.05$). A statistically significant difference was found in the Simple Programming Task dimension in the Scratch training ($p<0.05$). When the findings were evaluated, it can be said that there was a statistically significant increase in the complex programming skills of the students who received the Code.org training and in the simple programming task skills of the students who received the Scratch training after the training.

The pretest-posttest comparisons of the Children's Problem Solving Inventory for students in the Code.org and Scratch groups were examined using a Dependent Samples t-test and are presented in Table 10.

Table 10: Pretest-posttest comparison of the Children's Problem Solving Inventory of students

Group	Variable	Test	N	$\overline{X}$	S.s	sd	t	p
Code.org	Confidence in Problem Solving Skills	Pretest	15	43.40	6.84	14	-.426	.676
		Posttest	15	44.47	8.61			
	Self-Control	Pretest	15	25.27	6.09	14	-2.213	.044*
		Posttest	15	28.27	2.96			
	Avoidance	Pretest	15	19.33	4.13	14	-.514	.615
		Posttest	15	20.07	3.31			
Scratch	Confidence in Problem Solving Skills	Pretest	15	46.93	8.36	14	-.321	.753
		Posttest	15	47.60	7.01			
	Self-Control	Pretest	15	27.00	5.33	14	-1.532	.148
		Posttest	15	28.73	4.88			
	Avoidance	Pretest	15	20.87	2.90	14	-.816	.428
		Posttest	15	21.47	2.70			

*p<0.05

The students' pretest and posttest scores on the Children's Problem Solving Inventory before and after the Code.org and Scratch training were compared. Accordingly, a statistically significant difference was found in the Self-Control dimension in the Code.org training ($p<0.05$). However, no statistically significant difference was found in the Scratch training ($p>0.05$). When the findings were evaluated, it can be said that there was a statistically significant increase in the self-control skills of the students who received the Code.org training after the training.

The pretest-posttest comparisons of the Attitude Scale Towards Educational Computer Game-Supported Coding Learning of the students in the Code.org and Scratch groups were examined using a Dependent Samples t-test and are presented in Table 11.

Table 11: Pretest-posttest comparison of the Attitude Scale Towards Educational Computer Game-Supported Coding Learning of the students

Group	Variable	Test	N	$\overline{X}$	S.s	sd	t	p
Code.org	Desire to Learn Coding	Pretest	15	23.67	6.00	14	-2.689	.018*
		Posttest	15	27.87	8.61			
	Game Use	Pretest	15	29.80	5.85	14	-2.263	.040*
		Posttest	15	33.33	3.74			
	Effect of Computers on Socialization	Pretest	15	13.27	2.91	14	.751	.465
		Posttest	15	12.93	2.52			
Scratch	Desire to Learn Coding	Pretest	15	22.33	7.72	14	-.676	.510
		Posttest	15	23.60	8.29			
	Game Use	Pretest	15	30.07	8.09	14	-.995	.337
		Posttest	15	32.73	7.01			

Effect of Computers on Socialization	Pretest	15	14.33	2.79	14	.688	.503
	Posttest	15	13.73	3.03			

*p<0.05

The students' pretest and posttest scores on the Attitude Scale Towards Educational Computer Game-Supported Coding Learning were compared before and after the Code.org and Scratch training. Accordingly, a statistically significant difference was found in the Code.org training in terms of Enthusiasm for Learning Coding and Game Use (p<0.05). However, no statistically significant difference was found in the Scratch training (p>0.05). When the findings were evaluated, it can be said that the students who received the Code.org training showed a statistically significant increase in their enthusiasm for learning coding and their game use after the training.

Posttest comparisons of the Programming Self-Efficacy Perceptions of the students in the Code.org and Scratch groups were examined using an Independent Samples t-test and are presented in Table 12.

Table 12 Posttest comparison of students' self-efficacy perceptions of programming

Group	Variable	N	$\overline{X}$	S.s	sd	T	p
Simple Programming Task	Code.org	15	15.20	5.24	28	-.393	.697
	Scratch	15	15.80	2.73			
Complex Programming	Code.org	15	24.53	7.47	28	1.048	.304
	Scratch	15	21.47	8.53			

*p<0.05

When the students' post-test scores on their Perception of Self-Efficacy in Programming were compared after the Code.org and Scratch training, no statistically significant difference was found (p>0.05). Considering this, no significant difference was observed between the students' post-training scores, suggesting that the training provided was of similar levels.

The post-test comparisons of the Children's Problem Solving Inventory for students in the Code.org and Scratch groups were analyzed using an Independent Samples t-test and are presented in Table 13.

Table 13: Post-test comparisons of the Children's Problem Solving Inventory for students

Variable	Group	N	$\overline{X}$	S.s	sd	T	p
Confidence in Problem Solving Skills	Code.org	15	44.47	8.61	28	-1.093	.284
	Scratch	15	47.60	7.01			
Self-Regulation	Code.org	15	28.27	2.96	28	-.317	.754
	Scratch	15	28.73	4.88			
Avoidance	Code.org	15	20.07	3.31	28	-1.271	.214
	Scratch	15	21.47	2.70			

p<0.05

When the students' posttest scores on the Children's Problem Solving Inventory following the Code.org and Scratch training were compared, no statistically significant difference was found (p>0.05). Considering this, no significant difference was observed between the students' posttest scores, suggesting that the training provided was at a similar level.

Posttest comparisons of the Attitude Scale towards Educational Computer Game-Supported Coding Learning of students in the Code.org and Scratch groups were analyzed using an Independent Samples t-test and are presented in Table 14.

Table 14 Posttest comparisons of the Attitude Scale towards Educational Computer Game-Supported Coding Learning of students

Variable	Group	N	$\overline{X}$	S.s	sd	T	p
Desire to Learn Coding	Code.org	15	27.87	8.61	28	1.383	.178
	Scratch	15	23.60	8.29			
Game Use	Code.org	15	33.33	3.74	28	.292	.772
	Scratch	15	32.73	7.01			
Effect of Computers on Socialization	Code.org	15	12.93	2.52	28	-.785	.439
	Scratch	15	13.73	3.03			

$p<0.05$

When the students' post-test scores on the Attitude Scale Towards Educational Computer Game-Supported Coding Learning were compared after the Code.org and Scratch training sessions, no statistically significant difference was found ($p>0.05$). Considering this context, the students' post-training scores did not show a significant difference, suggesting that the training sessions were of similar quality.

Qualitative Findings

The post-training opinions of the students in the Code.org and Scratch groups are presented in this section. The distribution of codes and themes derived from the opinions of the students who received the Code.org training is presented in Table 15.

Table 15: Code distribution of the opinions of the students who received the Code.org training sessions

	Code.org	f	%
Sub-Themes	Liked Aspect	21	25,00
	Thought of Coding with Code.org	24	28,57
	Problems Encountered	15	17,86
	Opinions on Coding	24	28,57
	Total	84	100,00
Codes	Liked Aspect	f	%
	Composed of Different Sections	8	38,10
	Being Visual	6	28,57
	Being Gradual	2	9,52
	Having Tips	5	23,81
	Total	21	100,00
Codes	Coding Thought with Code.org	f	%
	Continuity	9	37,50
	Fun	1	4,17
	Favorite Application	12	50,00
	Thought of Being Limited	1	4,17
	Thought of Being Difficult	1	4,17
	Total	24	100,00
Codes	Problems Encountered	***f***	**%**
	Inadequacy of Explanations	1	6,67
	Block Placement	11	73,33
	Error Detection	1	6,67
	No Difficulty	2	13,33
	Total	15	100,00
Codes	Codes Opinions on Coding	***f***	**%**
	Progressive Difficulty	3	12,50
	Fun	12	50,00
	Enjoyable	3	12,50
	Easy	6	25,00
	Total	24	100,00

Four different subthemes were identified based on the students' opinions. Accordingly, the subthemes "Liked Aspect," "Thoughts of Coding with Code.org," "Problems Encountered," and "Opinions on Coding" were obtained. The distribution of codes and themes obtained based on the opinions of students who received Scratch training is presented in Table 16.

Table 16: Code distribution of opinions of students who received Scratch

	Scratch	*f*	%
Sub-Themes	Liked Aspect	13	17,57
	Problems Encountered	15	20,27
	Opinions on Coding	25	33,78
	Thoughts on Coding with Scratch	21	28,38
	Total	74	100,00
Codes	Liked feature	*f*	%
	Creating Visuals	11	84,62
	Enjoyable	1	7,69
	Being Online	1	7,69
	Total	13	100,00
Codes	Problems Encountered	f	%
	Order of Code	8	53,33
	Incomprehensibility of Menus	3	20,00
	Software Problem	2	13,33
	No Difficulty	2	13,33
	Total	15	100,00
Codes	Codes Opinions on Coding	f	%
	Fun	13	52,00
	Visual	4	16,00
	Enjoyable	4	16,00
	Easy	3	12,00
	Practical	1	4,00
	Total	25	100,00
Codes	Coding with Scratch Idea	*f*	%
	Suitable for Beginners	1	4,76
	Continuing Idea	10	47,62
	Enjoyable	3	14,29
	Favorite Application	6	28,57
	Restrictive Application	1	4,76
	Total	21	100,00

Based on the students' opinions, four distinct subthemes emerged. These subthemes were "Favorite Aspect," "Problems Encountered," "Views on Coding," and "Thoughts About Coding with Scratch."

Discussion and Conclusion

This study investigated the influence of a 10-week block-based programming intervention, utilizing Code.org and Scratch, on gifted fifth-grade students' problem-solving competencies, coding attitudes, and programming self-efficacy, while simultaneously examining their qualitative experiences. Synthesizing evidence from the mixed-methods design, the study establishes four primary conclusions. First, the sequential coding curriculum facilitated a statistically significant improvement in students' overall problem-solving skills, characterized by enhanced confidence and self-regulation, alongside a concomitant reduction in avoidance behaviors. Second, the intervention catalyzed a positive shift in students' attitudes toward computer game-supported coding. The gamified milestones of Code.org and the project-based structure of Scratch significantly bolstered enthusiasm for coding and social collaboration without precipitating technology saturation. Third, students' self-efficacy

regarding both fundamental and advanced programming tasks increased substantially; the structural transition from puzzle-oriented tasks to sandbox design successfully mitigated syntax-related anxiety and strengthened their perceived capability to manage nested logic. Fourth, qualitative findings indicate that gifted students perceived block-based programming as an intuitive and engaging process. Challenges encountered during complex scripting served as metacognitive triggers, effectively reframing "errors" from failures into constructive opportunities for debugging.

According to the findings of the study, there was no significant difference in the problem-solving skills of students who received coding training with Scratch, but there was a statistically significant difference in the self-control variable of problem-solving skills in students who received coding training with Code.org. In their study, Genç and Tınmaz (2010) stated that problem-solving skills did not improve because the structure of the programming language and the problems solved in the programming language course were not relevant to daily life. Bala (2019) stated that problem-solving skills did not improve because they could not apply the skills they learned at the practical level in new situations. Kalelioğlu and Gülbahar (2014) observed that teaching programming with Scratch had no effect on problem-solving skills, but they observed a slight increase in problem-solving abilities and self-confidence (Dizman, 2018; Yıldız, 2018). In their study, Mladenović, Krpan, and Mladenović (2017) stated that teaching programming with Scratch had no effect on problem-solving skills.

Dizman (2018) states that programming education positively impacts problem-solving skills and that the longer the programming education, the more positively it will be affected. Yıldız (2018) examined how problem-solving processes improve with Code.org in his study and noted that problem-solving skills are positively impacted. Ünsal (2020) suggests that using block-based coding tools like Code.org in younger age groups can enable students to acquire computational thinking skills at an early age, allowing them to easily solve problems they encounter in their future lives.

The significant increase in the self-control variable in the problem-solving inventory of students who received coding training with Code.org can be attributed to the fact that Code.org consists of sections, has informative videos at the beginning of each section, allows students to see when they make mistakes while coding and correct them, and allows students using Code.org to progress at their own pace (Astuti et al., 2024; Çalışkan, 2020; Karaduman, 2019; Yurtbakan, 2022). The reason for the lack of a significant increase in the confidence and avoidance variables in the problem-solving inventory of students who received coding training with Code.org is that gifted students are able to use their metacognitive skills effectively, thus easily solving the problems they encounter (Çağlar, 2022). The lack of a significant difference in coding training with Scratch has been interpreted as stemming from the lack of sufficient activities during the application phase while teaching basic programming concepts (Kalelioglu & Gülbahar, 2014; Shin & Park, 2014). When the findings of the study were examined, it was observed that the attitudes of students who received coding training with Scratch and Code.org changed positively towards coding. According to the pre-test and post-test results of the study, there was no significant difference in the Scratch group, but there was a significant difference in the Code.org group. Gülbahar and Kalelioğlu (2014) stated that students found the Scratch programming language easy and enjoyed writing and developing programs. Dinçer (2018) stated that although the attitude variable did not significantly increase after coding training with Scratch, students' attitudes towards programming increased with Scratch. These studies support the current study. Karaduman's

(2019) study found that Code.org had a positive impact on students' attitudes due to its dynamic and game-based nature. The significant attitudes of students who received coding training with Code.org toward coding can be attributed to Code.org's fun gameplay and dynamic nature (Choi & Choi, 2024). Although there was no significant difference in the attitude variable toward coding training with Scratch, interviews with students indicated that Scratch was easy and fun (Dinçer, 2018; Otu, 2020). This was interpreted as a positive development in their attitudes toward Scratch.

Students' scores on their Programming Self-Efficacy before and after the Code.org and Scratch training were compared. Accordingly, it can be said that there was a statistically significant increase in the complex programming skills of the students who received the Code.org training and in the simple programming task skills of the students who received the Scratch training after the training. It can be concluded that both block-based programming tools increased their perception of self-efficacy. Dinçer (2018) stated in his study that the ease of learning and use of the visual programming language, as well as its visual nature, increased their perception of self-efficacy. Derviş and Akcaoğlu (2026) stated that they enjoyed programming with Code.org, and as a result, their attitudes toward coding and their perception of self-efficacy increased. Yılmaz (2019) emphasized that Scratch, with its collaborative learning technique, could increase their perception of self-efficacy due to the high level of interaction and communication within the group. The study found that teaching students basic programming concepts with Scratch contributed to the development of simple programming skills (Aydoğdu, 2020; Kasalak & Altun, 2018). Code.org's modular structure, progression of sections from easy to more difficult, and students' ability to instantly identify and correct errors encountered in the program have all contributed to the development of complex programming skills (Korkmaz et al., 2021; Yurtbakan, 2022). These research findings can be interpreted as indicating that block-based programming tools enhance both simple and complex programming skills. The post-implementation opinions of gifted students regarding Scratch and Code.org were examined. Students who received training with Scratch and those who received training with Code.org stated that coding with both programming tools was easy, fun, and enjoyable. Students also expressed appreciation for the visual nature of both tools. Arslan and Akçelik (2019) stated that the ease of use and visual nature of visual programming tools would be effective for beginners. Çatlak, Tekdal, and Baz (2015) stated that using visual programming languages would be beneficial for understanding coding logic. Genç and Karakuş (2011) observed that programming instruction facilitated the acquisition of problem-solving processes, while beginners struggled with this. For this reason, they stated that using the Scratch programming language in teaching programming improves students' attitudes and facilitates programming learning. This supports the view that students do not have difficulty using this programming language. Yurtbakan (2022) stated that students found Code.org fun, which positively impacted their technology skills. Bala (2019) stated that coding with Scratch is fun and that attitudes towards the course will change. Karaduman (2019) stated that students found coding with Code.org easy and fun, emphasizing that it increases student achievement. Çavdar et al. (2022) stated that Code.org is visually advanced but lacks support and customization. This supports students' belief that Code.org is limited.

As a result, it is believed that using block-based programming tools for students new to programming will help them learn programming structures (Kyza et al., 2021). The visual nature of block-based programming tools, their ease of use, and their drag-and-drop logic make them a preferred method for students in the concrete operational stage to learn programming. The fun nature of coding with block-based coding tools has been interpreted as enabling students to easily learn programming logic while using these tools (Sungur &

Ozkan, 2026).

Recommendations

- A review of studies in the literature revealed that there are few studies using Scratch and Code.org in teaching programming to gifted students. More research is needed in this area. • This study examined the dependent variables "Problem-Solving Skills," "Attitudes Toward Educational Computer Game-Supported Coding Learning," and "Perceptions of Self-Efficacy." Different dependent variables can be studied.
- This study was conducted with 5th-grade students attending *BİLSEM*. The same study can be conducted with students at different educational levels attending *BİLSEM*.
- The study sample consists of 30 students. Studies with a larger sample size are possible.
- The study's implementation period is limited to 10 weeks, with one hour per week for a total of 10 hours. The total lesson hours can be increased in future studies.
- Future studies can target teachers who provide coding training using Scratch and Code.org.
- The effects of using different programming languages in coding education on students' problem-solving skills, attitudes toward coding, and self-efficacy can be examined.
- This study was conducted with students attending *BİLSEM*. The same study can be conducted with typically developing children.

Declarations

Funding

No funding was received from any individual or institution for this research.

Ethics Statements

The authors state that the research conducted with human participants was carried out in accordance with ethical principles and that the necessary ethical approvals were obtained.

Conflict of Interest

No potential conflicts of interest have been reported by the author(s).All authors have read and approved the final version of the manuscript.

Informed Consent

The authors confirm that informed consent was obtained from all individuals included in this study involving human subjects.

Data availability

Upon request to the corresponding author, the dataset associated with the research can be accessed.

Note

This article was produced from first author's MA thesis.

References

Akpınar, B., & Gezer, B. (2010). Reflections of New Learner-Centered Educational Approaches on the Learning-Teaching Process. *Dicle University Journal of Ziya Gökalp Faculty of Education*, *14*, Article 14.

Altun, A., & Mazman, S. G. (2012). Reliability and Validity Study of the Turkish Form of the Self-Efficacy Perception Scale for Programming. *Journal of Measurement and Evaluation in Education and Psychology*, *3*(2), Article 2.

Arslan, K., & Akçelik, M. (2019). Using Scratch in Programming Education: Attitudes and Perceptions of Pre-service Teachers. *Journal of the National Academy of Education*, *3*(1), Article 1. https://doi.org/10.32960/uead.455502

Astuti, Y. P., Subhiyakto, E. R., Umaroh, L., Sutojo, T., & Supriyanto, C. (2024). Pendampingan Penggunaan Media Pembelajaran Game Edukasi "Code.org" bagi Siswa SMP Ibu Kartini Semarang. *Abdimasku : Jurnal Pengabdian Masyarakat*, *7*(1), Article 1.

Avcı, E. (2021). *Determination of the effect of robotic coding activities designed for STEM education on gifted students' attitudes towards robotics and coding* [Master's Thesis]. İnönü University Institute of Educational Sciences: Malatya. YÖK National Thesis Center. (Thesis No. 659401).

Aydoğdu, Ş. (2020). The Effect of Block-Based Programming Activities on Pre-Service Teachers' Self-Efficacy Perceptions and Computational Thinking Skills Related to Programming. *Educational Technology Theory and Practice*, *10*(1), Article 1. https://doi.org/10.17943/etku.649585

Bakioğlu, A., & Levent, F. (2013). Suggestions for Turkey in the education of gifted students. *Journal of Gifted Education and Research (UYAD)*, 1(1), 31–44.

Bala, R. B. (2019). *Examining The Effects of Using Scratch Programming on 5th and 6th Graders' Problem Solving Skills* [Master's Thesis]. Necmettin Erbakan University Institute of Educational Sciences: Konya.

Balcı, H. (2024). The Effect of Block-Based Coding Activities on Text-Based Coding Activities. *Bucak Business Faculty Journal*, *7*(1), Article 1. https://doi.org/10.38057/bifd.1465228

Barradas, R., Lencastre, J., Soares, S., & Valente, A. (2020). Developing Computational Thinking in Early Ages: A Review of the code.org Platform: *Proceedings of the 12th International Conference on Computer Supported Education*, 157-168. https://doi.org/10.5220/0009576801570168

Becker, B. J., Aloe, A. M., Duvendack, M., Stanley, T. D., Valentine, J. C., Fretheim, A., & Tugwell, P. (2017). Quasi-experimental study designs series—paper 10: Synthesizing evidence for effects collected from quasi-experimental studies presents surmountable challenges. *Journal of Clinical Epidemiology*, 89, 84–91. https://doi.org/10.1016/j.jclinepi.2017.02.014

Bishara, A. J., Li, J., & Conley, C. (2021). Informal versus formal judgment of statistical models: The case of normality assumptions. *Psychonomic Bulletin & Review*, 28(4), 1164–1182. https://doi.org/10.3758/s13423-021-01879-z

Bozan, İ. (2022). *The effect of the 5e model supported by coding and digital game design on the academic achievement and problem-solving skills of gifted students* (Thesis No. 659401) [PhD Thesis]. Burdur Mehmet Akif Ersoy University Institute of Educational Sciences: Burdur. YÖK National Thesis Center. (Thesis No. 758421).

Bozan, I., & Taslidere, E. (2024). The effect of digital game design-supported coding education on gifted students' Scratch achievement and self-efficacy. *International Journal of Contemporary Educational Research*, 11(1), 20–28. https://doi.org/10.52380/ijcer.2024.11.1.531

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*, 77-101. https://doi.org/10.1191/1478088706qp063oa

Çağlar, Ç. (2022). *Secondary School 6th and 7th Grades. Investigation of Gifted Students' Metacognitive Awareness, Problem Solving Skills and Self-Efficacy Perception Levels* [Master's Thesis]. Mimar Sinan Fine Arts University Institute of Social Sciences: İstanbul. YÖK National Thesis Center. (Thesis No. 741566).

Calder, N. (2010). Using Scratch: An Integrated Problem-solving Approach to Mathematical Thinking. *Australian Primary Mathematics Classroom, 15(4), 9–14. https://search.informit.org/doi/abs/10.3316/INFORMIT.534282799471647*

Çalışkan, E. (2020). Investigation of the Effect of Code.org Activities on Secondary School Students' Problem Solving Skills and Programming Self-Efficacy. *Journal of Instructional Technologies and Teacher Education*, *9*(2), Article 2.

Çatlak, Ş., Tekdal, M., & Baz, F. (2015). The Status of Teaching Programming with Scratch Software: A Document Review Study. *Journal of Instructional Technologies and Teacher Education*, *4*(3), Article 3.

Çavdar, L., Kılıçer, K., & Emmioğlu, E. (2022). Code.org Evaluation of the online coding platform curriculum. *Journal of National Education*, 51(233), 689–714. https://doi.org/10.37669/milliegitim.799492

Choi, W. C., & Choi, I. C. (2024). *The influence and relationship between computational thinking, learning motivation, attitude, and achievement of Code.org in K-12 programming education* (arXiv:2412.14180). arXiv. https://doi.org/10.48550/arxiv.2412.14180

Derviş, M., & Akcaoğlu, M. Ö. (2026). Collaborative learning in programming education: Effects on self-efficacy and self-regulation among 6th grade students. *Journal of Computer and Education Research*, 14. https://doi.org/10.18009/jcer.1725535

Dinçer, A. (2018). *Comparison of Students' Attitudes, Self-Efficacy and Academic Achievement in Teaching Scratch and Kodu Game Lab Programming Languages to 6th Grade Students* [Master's Thesis]. Dokuz Eylül University Institute of Educational Sciences: İzmir. YÖK National Thesis Center. (Thesis No. 512965).

Dizman, A. (2018). *The effect of coding, robotics, 3D design and game design education on problem solving skills and metacognitive awareness level of 11-14 year old students* [Master's Thesis]. Bahçeşehir University Institute of Educational Sciences: İstanbul. YÖK National Thesis Center. (Thesis No. 523935).

Emiroğlu, M. (2021). *The effect of digital story on students' block-based coding achievements, attitudes towards coding education and activity perceptions in primary school coding education* [Master's Thesis]. Bahçeşehir University Institute of Educational Sciences: İstanbul. YÖK National Thesis Center. (Thesis No. 671522).

Erümit, A. K., & Şahin, M. A. (2025). Examining the Relationship Between Problematic Internet Use and Smartphone Addiction Levels and School Motivation among Middle School Students. *Journal of Uludag University Faculty of Education,* 38(2), 287-311. https://doi.org/10.19171/uefad.1549878.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). *Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study—ScienceDirect*. https://www.sciencedirect.com/science/article/abs/pii/S0360131512002813.

Fielding, N. G. (2012). Triangulation and mixed methods designs: Data integration with new research technologies. *Journal of Mixed Methods Research*, 6(2), 124–136. https://doi.org/10.1177/155868912437101

Geisinger, K. F. (2016). 21st century skills: What are they and how do we assess them? *Applied Measurement in Education*, 29(4), 245–249. https://doi.org/10.1080/08957347.2016.1209207

George, D., & Mallery, P. (2001). *SPSS for Windows Step by Step: A Simple Guide and Reference, 10.0 Update*. Allyn and Bacon.

George, D., & Mallery, P. (2016). *IBM SPSS Statistics 23 Step by Step: A Simple Guide and Reference* (14. bs). Routledge. https://doi.org/10.4324/9781315545899

Gülbahar, Y., & Kalelioğlu, F. (2018). Information and Communication Technologies and Computer Science: The Process of Curriculum Development *Journal of National Education*, *47*(217), Article 217.

Gürbüztürk, O., & Tanataş, D. Y. (2024). The Effect of Block-Based Coding Tools on Academic Achievement, Attitude and Computational Thinking Skills: A Meta-Analysis Study. *İnönü University Journal of Educational Sciences Institute*, *11*(21), Article 21. https://doi.org/10.29129/inujgse.1425193

Haymana, İ., & Özalp, D. (2020). The Effect of Robotics and Coding Education on Primary School 4. Creative Thinking Skills of Grade Students. *Istanbul Aydın University Journal of Faculty of Education*, *6*(2), Article 2.

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. https://doi.org/10.15388/infedu.2014.03

Karaduman, Ü. (2019). *The effect of the Code.org platform on programming learning of 6. Grade students.* [Master's Thesis]. Dokuz Eylül University Institute of Educational Sciences: İzmir.

Karaduman, Ü., & Akpınar, E. (2021). The effect of the Code.org platform on programming learning of 6th grade students. *Dokuz Eylül University Buca Faculty of Education Journal*, (51), 266–286. https://doi.org/10.53444/deubefd.832646

Genç, Z., & Karakuş, S. (2011). *Learning by Design-Using Scratch in Educational Computer Games Design | PDF*. https://www.scribd.com/doc/177523415/Tasar%C4%B1mla-O%C4%9Frenme-E%C4%9Fitsel-Bilgisayar-Oyunlar%C4%B1-Tasar%C4%B1m%C4%B1nda-Scratch-Kullan%C4%B1m%C4%B1

Karataş, H. (2021). 21st Century Skills İn Robotics and Coding Training Turkey and The. *Education and Society in the 21st Century*, *10*(30), Article 30.

Kasalak, İ., & Altun, A. (2018). Self-Efficacy Perception Scale Development Study for Block-Based Programming: Scratch Example. *Educational Technology Theory and Practice*, *8*(1), Article 1. https://doi.org/10.17943/etku.335916

Keçeci, G., Alan, B., & Zengin, F. K. (2016). Attitude Scale Towards Educational Computer Games Supported Coding Learning: Validity and Reliability Study. *Education Sciences*, *11*(3), Article 3.

Kılıç, S. (2013). Sampling Methods. (Turkish). *Journal of Mood Disorders*, *3*, 44-46.

Kırçalı, A. Ç. (2019). *Evaluation of computerized and non-computerized tools used in teaching algorithms at K12 level in terms of various variables* [Master's Thesis]. Marmara University Institute of Educational Sciences: İstanbul.

Kobsiripat, W. (2015). Effects of the Media to Promote the Scratch Programming Capabilities Creativity of Elementary School Students. *Procedia - Social and Behavioral Sciences*, *174*, 227-232. https://doi.org/10.1016/j.sbspro.2015.01.651

Koh, K. L., & Ahad, N. A. (2020). Normality for non-normal distributions. *Journal of Science and Mathematics Letters*, 8(2), 51–60. https://doi.org/10.37134/jsml.vol8.2.7.2020

Korkmaz, Ö., Balcı, H., Çakır, R., & Erdoğmuş, F. U. (2021). Effectiveness of Game Development Activities in Visual Programming Environments. *Mehmet Akif Ersoy University Journal of Faculty of Education*, *57*, Article 57.

Kurtdaş, M. (2023). Science and Art Centers in the Education of the Gifted and Gifted (Malatya Science and Art Center Example). *Thought – Comment Journal of Social Sciences Research. 5*, Article 10.

Kwak, S. G., & Kim, J. (2017). Central limit theorem: The cornerstone of modern statistics. *Korean Journal of Anesthesiology*, 70(2), 144–144. https://doi.org/10.4097/kjae.2017.70.2.144

Kyza, E. A., Georgiou, Y., Agesilaou, A., & Souropetsis, M. (2022). A cross-sectional study investigating primary school children's coding practices and computational thinking using ScratchJr. *Journal of Educational Computing Research*, 60(1), 220–257. https://doi.org/10.1177/07356331211027387

Lahullier, S. C. (2019). *Code.Org and Computational Thinking: A Mixed-Methods Study on Fifth-Grade Elementary School Students* [Ed.D.]. https://www.proquest.com/docview/2454660044/abstract/686AEC25A6B04215PQ/1

Mladenovic, M., Krpan, D., & Mladenovic, S. (2017). Introducing programming to elementary students' novices by using game development in Python and Scratch. *Edulearn16 Proceedings*, 1622–1629.

Newell, K. M., & Hancock, P. A. (1984). Forgotten moments: A note on skewness and kurtosis as influential factors in inferences extrapolated from response distributions. *Journal of Motor Behavior*, *16*(3), 320-335. https://doi.org/10.1080/00222895.1984.10735324

Okuducu, A. (2020). *The effect of Scratch supported mathematics teaching on 6. Grade students' academic achievement and attitudes about algebraic expressions* [Master's Thesis]. Ağrı İbrahim Çeçen University Institute of Science and Technology: Ağrı.

Oruç, Ş., & Çağır, S. (2020). Educational Expectations of Gifted Children. *Turkish Journal of Education*, *5*(2), Article 2.

Otu, T. (2020). *The effect of coding environments on secondary school students' achievement, attitude and computational thinking skills* [Master's Thesis]. Bolu Abant İzzet Baysal University Graduate Education Institute: Bolu.

Özçelik, A., & Akgündüz, D. (2018). Evaluation of Out-of-School STEM Education with Gifted/Gifted Students. *Trakya University Journal of Faculty of Education*, *8*(2), Article 2. https://doi.org/10.24315/trkefd.331579

Özmen, B., & Altun, A. (2014). Undergraduate Students' Experiences in Programming: Difficulties and Obstacles. *Turkish Online Journal of Qualitative Inquiry*, *5*(3), Article 3. https://doi.org/10.17569/tojqi.20328

Papadakis, S., Kalogiannakis, M., Zaranis, N., & Orfanakis, V. (2016). *(PDF) Using Scratch and App Inventor for teaching introductory programming in Secondary Education. A case study.* https://www.researchgate.net/publication/310733787_Using_Scratch_and_App_Inventor_for_teaching_introductory_programming_in_Secondary_Education_A_case_study

Saygıner, Ş., & Tüzün, H. (2017). *Difficulties in Programming Education and Solution Suggestions*. https://www.academia.edu/35416423/Programlama_E%C4%9Fitiminde_Ya%C5%9Fanan_Zorluklar_ve_%C3%87%C3%B6z%C3%BCm_%C3%96nerileri

Sayın, Z., & Seferoğlu, S. S. (2016). A new 21. Coding education as a century skill and the impact of coding on education policies. *Academic Informatics Conference*, *2016*, 3-5.

Scott, C. L. (2015). *The Futures of learning 2: What kind of learning for the 21st century? - UNESCO Digital Library*. https://unesdoc.unesco.org/ark:/48223/pf0000242996

Serin, O., Serin, N. B., & Saygılı, G. (2010). Development of the Problem Solving Inventory (CPCE) for Children at the Primary Education Level. *Primary Education Online*, *9*(2), Article 2.

Shin, S., & Park, P. (2014). A study on the effect affecting problem solving ability of primary students through the scratch programming. *Advanced Science and Technology Letters*, *59*(1), 117-120.

Sung, Y.-T., & Wu, J.-S. (2018). The Visual Analogue Scale for Rating, Ranking and Paired-Comparison (VAS-RRP): A new technique for psychological measurement. *Behavior Research Methods*, 50(4), 1694–1715. https://doi.org/10.3758/s13428-018-1041-8

Sungur, S., & Ozkan, G. (2026). Coding science: The role of block-based programming activities in enhancing computational thinking and science achievement. *Journal of Science Education and Technology*, 1-18.

Taş, N. (2018). *The effect of differentiated computer-aided mathematics activities on gifted students' computational thinking self-efficacy and attitudes towards mathematics* [Master's Thesis]. Atatürk University Institute of Educational Sciences: Erzurum.

Toklu, E. (2019). *Game Design and Coding Education for Gifted 9-11 Age Group Students - Kodu Game Lab Example* [Master's]. https://www.proquest.com/docview/2605205493/abstract/EEE03456AC564244PQ/1

Totan, H. N. (2021). *The effect of block-based coding education on secondary school students' computational thinking skills and attitudes towards coding learning: The case of Blocky* [Master's Thesis]. Necmettin Erbakan University Institute of Educational Sciences: Konya.

Tsai, M. J., Chien, F. P., Sun, W. T., & Jha, N. K. (2025). How block-based programming supports novice learners' coding comprehension: Evidence from eye-tracking lag-sequential analysis. *Computers & Education*, 105430. https://doi.org/10.1016/j.compedu.2025.105430

Ünsal, İ. (2020). *The effect of block-based programming activities on primary school 2. Grade students' computational thinking skills and activity perception* [Master's Thesis]. Sakarya University Institute of Educational Sciences: Sakarya.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends*, 60(6), 565–568. https://doi.org/10.1007/s11528-016-0087-7

Yıldız, S. (2018). *Examination of problem solving processes in block-based coding environment* [Master's Thesis]. Karadeniz Technical University Institute of Educational Sciences: Trabzon.

Yılmaz, Ş. (2019). *The effect of the use of co-learning technique on students' academic achievement and self-efficacy perception in Scratch program teaching* [Master's thesis, Afyon Kocatepe University]. YÖK Thesis Center.

Yükseltürk, E., & Altıok, S. (2015). Information Technologies Teacher Candidates' Views on Computer Programming Teaching. *Amasya University Journal of Faculty of Education*, *4*(1), Article 1. https://doi.org/10.16949/turkbilmat.399588

Yurtbakan, E. (2022). The Effect of Scratch Coding Program on Writing Tendencies, Concerns and Attitudes of Gifted Primary School Students. *Mersin University Journal of Faculty of Education*, *18*(3), Article 3. https://doi.org/10.17860/mersinefd.1130994

Zdawczyk, C., & Varma, K. (2023). Engaging girls in computer science: Gender differences in attitudes and beliefs about learning scratch and python. *Computer Science Education*. https://www.tandfonline.com/doi/abs/10.1080/08993408.2022.2095593