



Middle school students' views on mathematical situations designed with Scratch

Elif Çelik ^a , Sevim Sevgi ^{b*} 

^a Ministry of National Education, Yozgat, Türkiye.

^b Erciyes University, Kayseri, Türkiye.

Suggested citation: Çelik, E. & Sevgi, S. (2025) Middle school students' views on mathematical situations designed with scratch. *Journal of Educational Technology and Online Learning*, 8(2), 190-205.

Highlights

- The mathematical algorithms designed with Scratch developed positive attitudes in students.
- The majority of the students did not experience any difficulties.
- Students coding knowledge would contribute positively to their educational lives

Abstract

This research investigates the impact of mathematical problems created with Scratch on the problem-solving skills of 8th-grade middle school students. The research was based on the descriptive design of mixed research approaches. In this direction, the research model was single-group pretest-posttest design, which is a weak experimental design from experimental designs. Qualitative data were collected using the interview technique to complete the experimental design for the research. The participants were eight students studying in the 8th grade at Vocational Religious School at 2022-2023 academic year in central Anatolia, Turkey. Coding activities created with the Scratch program, developed by the researcher and containing mathematical situations, were carried out with the students. At the end of the activities, the data obtained from semi-structured interviews with 6 volunteer students were analyzed using the descriptive analysis technique. The mathematical algorithms designed with Scratch developed positive attitudes in students. The majority of the students did not experience any difficulties. Recommendations were made regarding the necessity of a strategic focus on the integration of computational thinking into the current curriculum in primary and middle school education and the evaluation of its effects on achievement.

Article Info: Research Article

Keywords: *Computational Thinking, Middle School Students, Problem-Solving Skills, Scratch*

1. Introduction

Polya (1945), one of the pioneers of problem solving, defined problem solving as finding a way out of a difficulty, getting around an obstacle, and reaching a goal that is not immediately understood. Schoenfeld (1983) defined a problem as an unsolvable concept, and that problem solving is just an exercise, no matter how difficult it is to solve. Green and Gilhooly (2005) expressed problem-solving as an activity that structures daily life meaningfully (as cited in Voskoglou and Buckley, 2012). The programming process helps students develop skills such as systematic and critical thinking and problem solving (Korkmaz, 2016; Lau and Yuen, 2011). The implementation of all processes from defining a problem to solving it on a computer screen refers to the problem-solving process through programming. Therefore, learning programming will also improve problem-solving skills (Hwang et al., 2008; Korkmaz, 2018). Problem-

* Corresponding Author. Department of Mathematics and Science Education, Faculty of Education, Erciyes University, Türkiye.

e-mail addresses: sevimsevgi@erciyes.edu.tr, clk.elif@hotmail.com

This article is a part of the first authors master's thesis which is "Scratch Kullanımının Matematiksel Problem Çözme Becerilerine Etkisi" under the supervision of second author, Assoc. Prof. Dr. Sevim SEVGİ

solving skills are the most complex of all mental skills (Gagné, 1985). It is a vitally important and metacognitive skill consisting of the steps of identifying the problem, gathering information, generating possible solutions, finding the appropriate strategy, and solving the problem (Mayer, 1998). Problem-solving skills are learned at a young age and developed during school age (Miller & Nunn, 2001); when combined with different disciplines, it increases students' problem-solving skills and students have better learning outcomes (McGehee, 2001). However, there are still uncertainties about how this will develop students' problem-solving skills in different courses with an interdisciplinary understanding and how this will be assessed. In the literature, problem-solving skills are often cited as an important benefit of creative coding activities in school-age children, but to date, very few studies have investigated how problem-solving can be used in interdisciplinary creative coding tasks (Woo & Falloon, 2022).

It has been nearly eighty years since it was first published, Polya's framework continues to have a global impact on our understanding of problem solving (OECD, 2013). Problem solving research is largely based on Polya's (1945) four-step model, which summarizes the basic stages of problem solving. This model consists of four stages: understanding, developing a plan, implementing the plan, and evaluating. Therefore, Polya's model also provides a suitable framework for understanding the general problem solving skills that can result from creative coding activities. Given the direct relevance of the research to computer science, it is appropriate to consider how Polya's problem solving and block-based coding were adopted. Taylor (2019) stated that Polya's problem solving model may be suitable for learning coding at the middle school level. Taylor (2019) stated that not all programming problems will require a level of problem solving; some problems may only require students to apply a known algorithm in the form of a common code pattern or may have simple algorithms that students can simply convert into code. Taylor (2019) described the steps to be followed when adapting the steps in Polya's problem solving to coding:

1. Understanding: The problem is clearly defined. Test scenarios are created based on the question text or the given test scenarios are examined. The person doing the coding can restate the problem in his/her own words or explain it to someone else.
2. Developing a Plan (Solving the problem without coding): Determining the level of problem solving required for the problem; This may be translating a given algorithm into code, applying a known algorithm example to the problem, or creating an algorithm to solve a problem. One of the various strategies to be used to create an algorithmic approach is to examine various examples to find a pattern that can be generalized to the algorithm. A complex problem is broken down into simpler problems, then each simple problem is solved separately. It is compared with alternative algorithms. An algorithm is selected and the solution is written in a way that is suitable for translating into code. No code is written at this stage yet.
3. Implementing the Plan (Translating the Solution into Code): In the second stage, the solutions determined are translated into code step by step. If the coder knows how to translate into code at this stage, he/she translates it himself/herself. Otherwise, the code is examined on the tool in which the program was made until it is understood how it works with support. The coding that was made is then run on a test input to make sure it works.
4. Evaluating (Testing the code): The code is run to ensure the functionality of all test cases. If a test case fails, debugging is used to identify and fix the algorithmic problem. The coder discusses and compares the solution with others to learn about alternative approaches to solve the problem.

2. Literature Review

Coding is a language used simply for the functioning of technology, requiring the breakdown of complex problems into parts and their reassembly in a logical order, thereby enhancing students' problem-solving skills; learning this language equips students with skills such as task sequencing and expressing their ideas in creative ways, in addition to problem-solving (Siegle, 2017). Woo and Falloon (2022) stated that not all coding problems require students to produce an original solution; some problems only require students to apply known coding patterns or convert a known algorithm into code. In such problems, students can use pattern recognition methods when applying known coding patterns to the given programming problem or recognizing patterns in data. They will engage in the algorithmic thinking process when creating a series

of step-by-step instructions. Algorithms for simple problems can be obvious. Woo and Falloon (2022) stated that not all programming or coding problems may require all stages of computational thinking, and that there may be alternative methods that students can use to solve programming or coding problems.

Students need to develop problem-solving skills at an early age (Papert, 1980). Computer programming is an ideal tool for developing these skills. Coding can fundamentally change our perspective on technology and transform us from passive consumers into active producers (Morrison, 2013). Block-based programming is an ideal method for beginners. Due to its ease of understanding, students quickly start coding, minimize user errors, and acquire the necessary foundations for transitioning to text-based coding (Siegle, 2017). Coding education can contribute to children's cognitive and social development by enhancing their computational thinking, creativity, and problem-solving skills (Read and Macfarlane, 2006). Students who receive coding education have shown significant improvement in their mathematics performance and social skills (Wong et al., 2015). It holds an important place in the coding education system. However, it is still not fully understood how different technological systems play a role in developing computational thinking and problem-solving skills (Zhu et al., 2016). More research is needed on this subject. It is important to determine which cognitive skills different systems are more effective in developing and which children can benefit more from which systems. There is evidence that using concrete input enhances children's performance in robotic programming (Sapounidis et al., 2015). How different combinations of inputs and outputs affect problem-solving skills has not yet been sufficiently researched. This topic is an important area that should be the focus of future research. Therefore, the impact of coding activities on mathematical problem-solving skills has contributed to the literature.

2.1. Mathematics and Coding

In research linking coding to mathematics, Howard and Howard (2020) state that the origins of coding come from mathematics and that the relationship between them naturally exists. There is research indicating that coding guides students on how to practically use skills such as mathematics, algorithms, problem-solving, and analysis (Fessakis et al., 2013; Israel et al., 2015; Jona et al., 2014). Falloon (2016) stated that coding helps students improve their problem-solving abilities and enhance their higher-order thinking skills. Byrne and Lyons (2001), while examining the relationship between mathematics and coding, state that programming skills bear similarities to mathematical problem-solving skills. Calder (2010) stated that coding can provide motivation in the learning process of mathematical processes and skills. Calder (2010) stated that coding using Scratch helps students develop various mathematical concepts and contributes to the further enhancement of skills such as problem-solving, logic, and reasoning.

In research linking coding to mathematics, Howard and Howard (2020) state that the origins of coding come from mathematics and that the relationship between them naturally exists. There is research indicating that coding guides students on how to practically use skills such as mathematics, algorithms, problem-solving, and analysis (Fessakis et al., 2013; Israel et al., 2015; Jona et al., 2014). Falloon (2016) stated that coding helps students improve their problem-solving abilities and enhance their higher-order thinking skills. Byrne and Lyons (2001), while examining the relationship between mathematics and coding, state that programming skills bear similarities to mathematical problem-solving skills. Calder (2010) stated that coding can provide motivation in the learning process of mathematical processes and skills. Calder (2010) stated that coding using Scratch helps students develop various mathematical concepts and contributes to the further enhancement of skills such as problem-solving, logic, and reasoning. Research indicates that learning to code can be challenging for many students. Ben-Ari (1998) stated that when learning to code, students may find it challenging to encounter new words, concepts, programming languages, and a completely new approach style. Malan and Leitner (2007) suggest that coding is like learning a new language and has a completely different language structure. Concepts such as loops, variables, algorithms, Boolean expressions, and conditional logic are often unfamiliar to students who are new to coding (Malan and Leitner, 2007). Another barrier to coding is that while most students frequently use technology in their daily lives, they may not possess the skills to use technology efficiently (Howard and Howard, 2020). Kafai and Burke (2013) state that students may have the ability to use technology, but they may not always be able to use this skill for purposes such as creativity, critical thinking, or productivity. Korkmaz (2016) stated

that there is no existing research on the concrete results of the basic programming education given to students through Scratch in later periods.

Howard and Howard (2020) stated that coding promotes skills such as critical thinking and creativity; it is a cognitive process that tends to use reasoning, logic, and inference skills. Korkmaz (2016) states that, contrary to the view that coding exists solely for programming, it also emphasizes conceptual and design-related problem-solving skills. Chao and Cheng (2000) reported that Piaget stated that deductive reasoning is a cognitively advanced skill that develops during adolescence; particularly, adolescents acquire a complete mental logic corresponding to standard logic during this period. Howard and Howard (2020) stated that students might therefore struggle to apply logic and reasoning skills in coding. Altun et al. (2016) stated that students might struggle due to their attention being easily diverted in a short period and having to work on multi-step activities when writing code. Howard and Howard (2020) stated that mathematics and coding can be seen as two closely related but distinct fields, and that the concept of computational thinking can serve as a bridge between these two areas. They noted that teaching both mathematics and coding together can help develop a strong mathematics and coding program.

In their research using the multiple-choice test method to evaluate coding activities conducted with Scratch, Ericson and McKlin (2012) used a multiple-choice test to measure the learning of their students in coding activities that included the use of Scratch software for grades 4-12. In the 10-question multiple-choice test they used to measure students' learning, they expected students to establish a relationship between the problem situations presented and the code blocks they used throughout their education, and this relationship was examined question by question. Franklin et al. (2017) used the test method to measure learning using Scratch.

2.3. Scratch and Computational Thinking

One of the most common applications of computational thinking is coding, and not involving all children in the coding process can harm the potential for participation in the structures that will be built in the field of coding in the future (Kotsopoulos et al., 2017). It would be wrong to consider coding merely as a part of programming. Because Papert (1980) argues that computational thinking is a good way to develop other thinking skills. Papert's (1980) definition of computational thinking includes solving appropriate problems with computational ideas and computer literacy. Wing (2006) defines computational thinking as the thought processes involved in formulating problems and solutions in a form that can be represented. The programming language consists of abstract expressions. This language can lead students to perceive the programming process as a phenomenon that does not align with real situations (Korkmaz, 2016). Environments like Scratch and Logo are among the tools that transform the abstract nature of programming into a more observable and concrete form for students (Korkmaz, 2016). Coding can also be used in different subjects with an interdisciplinary approach. For example, students can explore pre-prepared programs to discover how mathematics is used in coding. These programs can help students better understand how mathematics integrates with coding by focusing on the lines of code where mathematics is most prominent. Students can discover that by focusing on variables and mathematical formulas and adjusting the code, they can instantly observe the results of their changes, revealing that coding is a new way to experience, represent, and investigate mathematics. In this way, they can grasp mathematical concepts more concretely and reinforce their mathematical thinking while developing their coding skills (Gadanidis, 2015). From a pedagogical perspective, focusing on the "if" question provides a context for making assumptions, problem-solving, and generalization (Kotsopoulos et al., 2017).

In some studies, there are systems that use Web 2.0 technology to enable students to collaborate with each other, inspire one another, and gain some educational benefits from collaborative learning, but it is not clear which skills are acquired in the process (Ioannidou et al., 2011). When programming through coding, the computational thinking potentially learned (Yalçinkaya et al., 2018) can be argued to represent basic-level programming competence rather than high-level computational thinking, as the code structures used with tools like Scratch. When using tools like Scratch, the programming activities students engage in can develop skills and areas of understanding related to computational thinking in different ways. However, the practical programming opportunities in Scratch have not been systematically and comprehensively researched in the

literature to understand how they can represent and develop various skills and areas of understanding related to computational thinking, as defined in basic concepts and applications (Fagerlund et al., 2021). Yılmaz and İzmirli (2023) stated that, Scratch block-based coding activities, did not show a significant change in perception of computational thinking skills in the post-test compared to the pre-test. However, their performance in computational thinking skills improved significantly in the post-test compared to the pre-test. Therefore, it would be appropriate to evaluate Scratch and the functions of the block library in terms of computational thinking concepts. Brennan and Resnick (2012) address computational thinking in three dimensions: concepts of computational thinking, applications of computational thinking, and perspectives of computational thinking. Computational thinking concepts include fundamental computer science principles such as algorithms and data structures. Computational thinking applications emphasize how these concepts are used in problem-solving and project development, while computational thinking perspectives examine how individuals and societies perceive technology and computing and how they interact with them.

3. Method

3.1. Research Model

We employed the explanatory design, a mixed design that integrates qualitative and quantitative methods. According to this design, researchers collect qualitative data to complete and refine it (Büyüköztürk, 2023). We collected qualitative data using the interview technique. We used the single-group pretest-posttest model, a weak experimental design. Weak experimental designs do not involve random assignment or group matching. In the single group pretest-posttest model, a pretest and posttest are given to a randomly chosen group before introducing the independent variable (Yıldırım and Şimşek, 2021). The research model is given in Table 1.

Table 1.

Single-Group Pretest-Posttest Model

Group	Pre Test	Intervention	Post Test
Single Group	Pre test	Coding activities	Post test

3.2. Research Group

The research group consisted of eight eighth-grade female students studying at a religious vocational middle school in a city center in Central Anatolia. Since it is a vocational religious school, only female students are studying. All students in the eighth grade currently in the school were reached. Following the implementation of the single-group pretest-posttest model, six volunteer students were interviewed.

The stratified purposive method was used, which is a non-random sampling method. This method allows for the identification, description, and comparison of specific subgroups and provides the opportunity to work with easily accessible and extreme subjects instead of random selection (Büyüköztürk, 2023). 20% of the eighth-grade students studying in the neighborhood were reached. We selected the participants from eighth-grade students whose abstract thinking and problem-solving skills were in the development stage. This selection was based on the assumption that an activity aimed at coding and problem-solving with Scratch could contribute to the development of skills.

3.3. Data Collection Tool-Semi-Structured Interview Form

The pretest was first applied to eight students studying in the eighth grade. The posttest was applied again after the completion of the coding training. As a result of the research, interviews were also conducted with six volunteer students. Expert opinions were obtained while preparing the interview questions. You can read the details of interview form at the Çelik (2024)'s study.

A semi-structured interview form was used to determine the students' views on the block-based coding activity. An interview is a communication process that is planned in advance, carried out for an important goal, focused on asking and answering questions, and includes mutual interaction (Karataş, 2015). The reason for using the semi-structured interview technique is that it is based on a predetermined format and

provides flexibility, allowing the researcher to provide more organized and comparable information (Yıldırım and Şimşek, 2021).

The interview form was prepared by the first researcher after receiving the opinions of two field experts in order to determine the students' opinions about the coding activity. The interview questions were prepared twice and finalized after being presented to the opinions of two experts. The interviews in question were conducted by the first researcher, and the following questions were asked:

1. What do you think about the coding training you received? Have you had any coding experience before?
 2. What do you think about the importance of the coding training you received? If so, can you explain in what way it is important?
 3. What are the difficulties you encounter while coding?
- Sub-Question: Do you have a coding workshop at your school where you can receive coding training?
4. How can having coding skills affect your education or your future career choice?

Sub-Question: Do you see coding as an important skill that every student should have today? Why?

5. Do you think there is a relationship between math and coding? If the answer is yes, “Can you explain this relationship?”
6. Did the coding training you received affect your interest in math? Explain why in detail.
7. Do you think the coding training you received can be used in different courses? Explain why by giving examples.

3.4. Data Collection

A pretest was applied to determine the problem-solving skills of eighth graders, and then coding training was provided via Scratch. The activities given in Table 2 were applied in this training. The activities were determined according to the age groups of the students and were designed in accordance with the course content and objectives. At the same time, they were prepared by taking into account the problem situations that can be applied in daily life. In the process of the application to be done, Scratch was introduced to the students, and the functions of the code blocks found were explained.

The functions of code blocks and sample coding activities were presented to the students.

1. They were asked to create activities suitable for the purpose.
2. After the coding activities, students were interviewed to get their opinions.



The real names of the participants were hidden and code names such as K1, K2, etc. were used. In the interviews with the students included in the research group, students' expressed their opinions freely. The conversation took place in a friendly environment and the students were comfortable.

A scene consisting of sample programs with structures called puppets and coding-based scenarios to be constructed in this scene was developed to demonstrate the concepts of introductory Scratch programming. These scenarios were designed to serve as potential starting points for students' projects. Students were presented with various activities appropriate to their current curriculum. Activities were listed according to the complexity of the code blocks to be used for coding, not in the order of objectives. The general progress of the lessons, including the emphasis on coding and the mathematical concepts introduced, and the objectives of the activities (MoNE, 2023) are summarized in Table 2.

Students were interviewed after the completion of the coding activities. Students were volunteers. The interviews were conducted in a semi-structured format. At the beginning of the interviews, participants were informed about the purpose, process, and confidentiality principles of the research. Participants were asked to share their experiences, opinions, and suggestions regarding the research in line with the questions in the semi-structured interview form. During the interviews, participants' statements were noted, and questions were asked when necessary.

Table 2.

Scratch Activities, Implementation, and Problem-Solving Process

Activity No And Name	Objective	Problem-solving process (Polya, 1945; Taylor, 2019)	Scratch Page
1. Question and Answer	Short question-and-answer activities are used in dialogues between sprites.	<p><u>Defining the Problem:</u> (The student)</p> <ul style="list-style-type: none"> determines the questions to ask. The questions are related to the objectives listed in the objectives column. knows the answers to the questions they will ask. determines the backdrop, sound, and sprites they will choose. <p><u>Designing and Implementing a Plan:</u> (The student)</p> <ul style="list-style-type: none"> (Applies code templates and uses code sequences according to the problem-solving level) The code blocks to be used are determined. assigns variables. uses the "Ask" and "Answer" code blocks. uses the "If" block to provide feedback for correct or incorrect answers. <p><u>Evaluation:</u> (The student)</p> <ul style="list-style-type: none"> clicks the "start" button. checks if the code is working. 	
2. Is It Prime?	Codes a program using Scratch code blocks to determine if a number is prime.	<p><u>Defining the Problem:</u> (The student)</p> <ul style="list-style-type: none"> knows how to determine if a number is prime. determines the backdrop, sound, and sprites they will choose. <p><u>Designing and Implementing a Plan:</u> (The student)</p> <ul style="list-style-type: none"> (Applies code templates and uses code sequences according to the problem-solving level) The code blocks to be used are determined. uses the "Ask" code block. assigns variables. uses the "Modulo" block to check for divisors of the number. uses conditional blocks to exclude 1 and the number itself. If no divisors are found, the program uses the "Answer" code block to display the number as "prime." <p><u>Evaluation:</u> (The student)</p> <ul style="list-style-type: none"> clicks the "start" button. checks if the code is working. 	

3. Positive Integer Divisors

Codes a program with Scratch code blocks to find the positive divisors of a positive integer.

Defining the Problem: (The student)

- knows how to calculate the positive integer divisors of a number.
- determines the backdrop, sound, and sprites they will choose.

Designing and Implementing a Plan: (The student)

- (Applies code templates and uses code sequences according to the problem-solving level) The code blocks to be used are determined.
- uses the "Ask" code block.
- assigns variables.
- uses the modulo block to check all numbers up to half of the given number.
- separates the numbers with a remainder of zero using "variable" blocks and lists them using "list" blocks.
- uses the "List" block and the "Answer" code block to display the "positive divisors" of the number.

Evaluation: (The student)

- clicks the "start" button.
- checks if the code is working.



4. Are They Coprime?

Codes a program using Scratch code blocks to determine if two numbers are coprime.

Defining the Problem: (The student)

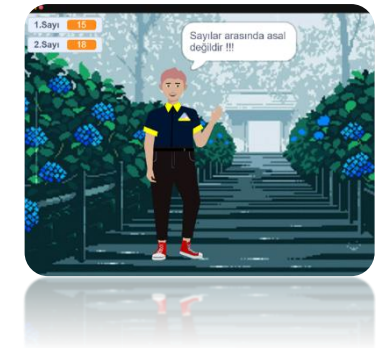
- knows how to determine if two numbers are coprime.
- determines the backdrop, sound, and sprites they will choose.

Designing and Implementing a Plan: (The student)

- (Applies code templates and uses code sequences according to the problem-solving level) The code blocks to be used are determined.
- uses the "Ask" and "Answer" code blocks.
- assigns variables.
- utilizes "operator" code blocks to determine the smaller of two numbers inputted into the program via the "answer" code block.
- employs the "modulo" block to divide all numbers up to the smaller number, checking for GCD.
- uses the "answer" code block to have the program display whether the numbers are "coprime" or "not coprime."

Evaluation: (The student)

- presses the "Start" button.
- verifies the functionality of the code.



5. GCD and LCM Calculation Program

Develops a program using Scratch code blocks to calculate the GCD and LCM of two entered numbers.

Problem Definition: (The student)

- understands how to calculate GCD and the Least Common Multiplies (LCM) of two numbers.
- selects the backdrop, sound, and sprites.

Designing and Implementing a Plan: (The student)

- (Applies code templates and utilizes code sequences based on the problem-solving level) The student determines the code blocks to be used.
- uses the "ask" code block.
- assigns variables.
- utilizes the "modulo" block to check if the variable k divides both input numbers evenly.
- isolates numbers where the remainder is zero using "variable" blocks; isolates numbers meeting the condition using "conditional" blocks.
- employs "conditional" blocks for exceptional cases.
- uses the "answer" code block to display the GCD and LCM values.

Evaluation: (The student)

- clicks the "start" button.
- verifies the functionality of the code.



3.5. Data Analysis

We used the content analysis technique to analyze the data from semi-structured interviews. The main goal of descriptive analysis is to reach concepts and relationships that can make sense of the collected data (Şeker, 2020). Descriptive analysis is a type of qualitative data analysis that involves summarizing and interpreting information obtained through various data collection methods according to certain themes (Gültepe, 2018). In the analysis, the statements of the interviewed students were conveyed with direct quotes, and the findings were presented in an organized and interpreted manner. In content analysis, the aim is to reach concepts and relationships related to these concepts that can explain the collected data. In descriptive analysis, the summarized and interpreted data are processed in more detail during the content analysis process; thus, concepts and themes that cannot be recognized with a descriptive approach may emerge. For this reason, content analysis is performed by first conceptualizing the collected data, then organizing it logically, and classifying it according to themes (Yıldırım and Şimşek, 2021).

3.6. Validity and Reliability

The use of interview questions from the literature is crucial for ensuring the internal validity of the test. External validity was established by linking the coding activities to students' mathematical performance, assuming that the activities developed mathematical computational skills. It was assumed in the study that consistency was ensured by administering interview questions. Some factors that may pose a threat to the validity and reliability of interview questions and the precautions taken by the researcher against these factors are provided below:

External Threats:

Attacks on Test Security: During the interview, malicious individuals or groups from external sources may compromise the security of the interview questions. There were no instances where students obtained interview questions or answers beforehand.

Interference with the Examination Process: No situations occurred where external threats created technical problems during the interview, preventing students from completing the test normally.

Internal Threats:

Recalling Answers: There is a risk that students might memorize the answers from previous interviewed colleagues. Especially in interviews repeated within a short period, the probability of students' memorization the answers may increase. To address this threat, the researcher individually cared with each student and requested not to share the interview questions by examining the students' approaches to the questions, student answers were shared in some places.

Threatening Behaviors: Student behaviors that could affect the reliability of the test are also among the internal threats. There were no attempts by students to cheat or engage in fraudulent activities.

4. Findings

4.1. Positive Aspects of Mathematical Situations Designed with Scratch

Table 3 presents the themes and codes related to students' views on the positive aspects of designing mathematical algorithms with Scratch. According to the students' opinions, the mathematical algorithms designed with Scratch developed positive attitudes in students; they found the activities useful, it increased their interest in the lesson, they had fun while coding, they developed their sense of achievement, they saw coding as a profession of the future, it helped them to eliminate their deficiencies in the subject and to reinforce the subjects, they gained experience about coding, the activity increased their self-confidence in coding, and it raised awareness about coding. They also stated that using coding

in different lessons would make the lessons more engaging. Below are direct quotes that express the students' opinions on the subject.

Table 3.

Advantages of Mathematical Algorithms Created with Scratch

Codes	Theme Codes (Sub Themes)	Student	f	%
Usefulness	Useful	K1, K6	2	33
	Fixing subject deficiencies	K3	1	16
	Entertaining	K2, K4	2	33
	Seeing importance for the future	K1, K2, K5, K6	4	66
	Positive attitude	K1, K2, K3, K4, K5, K6	6	100
Development	Increased interest in the course	K5	1	16
	Sense of achievement	K1, K2, K5	2	50
	Improved logic and reasoning	K6	1	16
	Gained experience	K4	1	16
	Increased self-confidence	K1	1	16
	Awareness about coding	K4, K5	2	33
Career Choice	Software Engineering	K1, K6	2	33
	Mathematics Teaching	K2	1	16
	Does not affect	K3, K4, K5	3	50
Interdisciplinary	Use of coding in different courses	K2, K3, K4, K5	3	50

K1: Coding training was good. I'm already thinking of becoming a software developer when I grow up, so if I win the scholarship, I'm thinking of saving my money and buying a computer. I'm thinking of saving my money, buying a computer, and learning. This gave me confidence in my ability to achieve my goals.

K2: It was very enjoyable, wonderful, and also more memorable to learn things while having fun. (Do you see coding as a skill that every student should have?) Yes, I do because we are entering a more technological era. It also affected our success in math. Math was difficult before. Math is a little more fun now. (About using coding in different classes) Using coding in traditionally dull classes could make them more engaging and enjoyable. The class would attract more attention.

K3: It was very fun and educational. It helped me reinforce the topics. (About the effect of coding on career choice and education) I don't know what to say. If I become a teacher in the future, I will have gained knowledge. I want to be a math teacher. (What impact did the coding education you received have on your interest in math classes?) I already liked the class; it was more enjoyable. (About using coding in different classes) I would like it. It would be useful. It would make those classes more enjoyable, too.

K4: It's a wonderful activity. Coding training is important because we learn in a more fun way. (What is the effect of coding on career choice and education) It can be. I don't think coding will affect my career choice, but it will influence my educational path. For example, if we do coding in high school, I will be more aware of its importance.

K5: I think it has a positive effect. There were some difficulties, though. (About the importance of coding training) I mean, it's important. It increased our knowledge. (About the coding training you receive affecting your interest in math) I already like math, but my performance is low. It increased my math performance a little bit. It increased my love for the mathematics. (About the use of coding in different courses) It should be used. You can use in the revolution course. I don't like that course very much. It has the potential to transform me into a passionate student.

K6: I think (coding training) is useful because I answer within a certain logical framework, which strengthens my logical skills. For example, I realized that I answered the questions in the test you gave by embracing them more. It was also fun in terms of feelings. I was considering pursuing a career in software engineering. So I think it will contribute at least a few things. (About whether or not coding is a skill that every student should have) If this is a talent, it is not appropriate to expect perfect success from every

student, of course. However, I believe that students should receive at least some training in this area. They should at least experience it. Maybe they are very talented. Discovering their talent might be crucial. (About the relationship between coding and mathematics) In other words, mathematics is based on formulas. I think coding is the same. (About the effect of coding on their interest in the course) It had a positive effect. After coding, I realized what the questions meant and that I had made the connection between the operations faster. It increased my self-confidence because I made the connection between the operations faster.

4.2. Negative Aspects of Mathematical Situations Designed with Scratch

The themes and codes belonging to these themes, which include the views of students who designed mathematical algorithms with Scratch regarding the negative aspects of the coding activities they carried out, are given in Table 4. According to the students' views, the internet outage and difficulties with the Mathematical Algorithms Designed with Scratch application were identified as problematic aspects. The majority of the students did not experience any difficulties. They were unsure about coding's use in various fields and said no because they didn't know how to use it. Below are direct quotes expressing the students' perspectives on the subject.

Table 4.

Disadvantages of mathematical algorithms created with Scratch

Codes	Theme Codes (Sub Themes)	Student	f	%
Insufficient infrastructure	Internet outage	K5	1	33
	Lack of information	K2	1	33
No Negative Aspects		K1, K3, K5, K6	4	66
Interdisciplinary	Use of coding in different courses	K1, K6	3	50

K1: (About the difficulties he encountered while coding.) No difficulty. Simple. (About the use of coding in different courses) So there is no need to do coding specifically.

K2: Sometimes I can't understand what I need to do while coding. I have difficulty because I don't know how to put what I think into coding.

K3: (About the difficulties he encountered while coding.) None. Some stages were challenging but manageable. Placing the codes was sometimes difficult.

K4: (About the difficulties he encountered while coding.) Like, sometimes we code, and it doesn't save.

K5: (About the difficulties he encountered while coding.) I didn't have much difficulty. We couldn't do it on the phone. There were internet interruptions. We had to start coding from the beginning. We got stressed.

K6: (About the difficulties he encountered while coding.) I don't think I had that much difficulty. (About the use of coding in different courses.) I don't think it can be added to courses that are not very related to the things coding develops. We can concentrate on courses that necessitate a higher level of logic, reasoning, and skills. Since you apply more logic and reasoning in mathematics, I think the connection between operations has a greater impact on mathematics.

4.3. Difficulties of Mathematical Situations Designed with Scratch

Themes and codes belonging to the themes that include the views of students who designed mathematical algorithms with Scratch about the incomprehensible aspects of the coding activities they carried out are presented in Table 5. None of the students could answer the question "Do you have a coding workshop?" because they did not understand what a "coding workshop" is. They said no because they didn't know how to use coding in various fields.

Table 5.

Incomprehensible aspects of mathematical algorithms created with Scratch

Theme	Codes / sub themes)	f	%
Misconception	Coding Workshop	6	100

As a result of the interviews conducted with the students, the majority of the students found the activities enjoyable and developed a belief that the coding knowledge they acquired would contribute positively to their educational lives. Some students had issues, like not fully grasping the code blocks' functions or saving their projects, but they had no trouble with the activities. Despite their full participation in the activities, not all students completed every project, either by leaving it half-done or not starting at all. Table 6 displays the percentages of students who completed the activities.

Table 6.

Completion rates of activities

Activity	Completed	%
Question-Answer	4	50
Is it Prime?	6	75
Positive Integer Divisors	8	100
Are they Co-Prime?	5	62.5
GCD and LCM Calculation Program	6	75

As seen in Table 6, all students completed the positive divisor number coding activity, and the lowest completion rate was the question-answer activity. This situation may be because question-answer activity was the first coding activity performed, and the students experienced problems such as saving the code blocks and the interface of the application due to their new acquaintance. The fact that some projects were left unfinished by the students may have resulted from a situation such as the student that the activity. The student may not have understood the activities or lacked the motivation to start them.

4. Discussion, Conclusion, and Suggestions

Most of the students showed positive attitudes toward programming with Scratch and using it in mathematics classes. Korkmaz (2016) found that Scratch-based educational activities did not have any effect on students' attitudes towards programming education compared to traditional methods. This is consistent with Genç and Karakuş's (2011) findings that students' attitudes towards using Scratch were positive (Erol and Çırak, 2022; Genç and Karakuş, 2013); however, it is inconsistent with Korkmaz's findings (2016). In addition, in our study, students stated the unnecessary need to take coding education as a separate course. This statement partially coincides with Korkmaz's (2016) findings that students found programming education unnecessary.

When asked how students' perceptions of their education and future career choices are affected, we received answers that they could not say for sure; they felt a positive effect, but how this effect will affect their education and training lives, and that it will not affect the majority of them. Ioannidou et al. (2011), found in their research that many students, especially girls in high school, think of computer science as just programming, which is one reason why they show little interest in the subject. This research was part of the iDREAMS project (Integrative Design-based Reform-oriented Educational Approach for Motivating Students) funded by the National Science Foundation which looked at how teaching methods like Scalable Game Design can spark more interest in computer science. The students' view that it will not affect their career choices can support this view of Ioannidou et al. (2011).

Scalable game designs provide students with a sense of accomplishment and the opportunity to transition to computer science and STEM content (Repenning, 2006). This is consistent with our findings that some students saw that they could do the activities after completing them, and this increased their desire to become software engineers in the future. While Scratch created a difference in algorithmic, creativity, and critical thinking skills in the experimental group, there was no significant mean difference between the

experimental and control groups in terms of collaborative and problem-solving skills (Bakır, 2023). We also found that it did not create a significant mean difference in problem-solving skills.

Since the 1990s, studies on increasing student motivation in computer science education have been successful (Werner et al., 2012). However, the effects of these studies on student achievement are unknown (Lye & Koh, 2014). One of the problems encountered is the lack of appropriate tools to measure learning in the classroom (Lye & Koh, 2014). Existing tools, such as rubrics, can be used to measure learning. However, these tools are time-consuming and have limited functionality in terms of tracking learning progress and providing continuous feedback to students and teachers (Lye & Koh, 2014). Our interviews with students and the findings of the pre-test and post-test support these statements. We can say that mathematical algorithms coded with Scratch increase students' motivation and interest in the course, but it is obvious that existing measurement and evaluation tools are inadequate in terms of reaching accurate judgments about what and to what extent students have achieved. This situation may prevent students from fully realizing their learning potential.

Students finding the coding activities fun increases their interest and motivation in the course, but it is not a prerequisite for academic success. More research is needed on how coding can be integrated into courses with an interdisciplinary approach in a way that simultaneously increases students' motivation and success.

References

- Alp, G. (2019). *Scratch programı ile web destekli işbirlikli öğrenme yönteminin ilköğretim 5. sınıf öğrencilerinin kavramsal anlama düzeylerine ve eleştirel düşünme becerilerine etkisi* (Thesis number. 543479) [Master's Thesis, Bursa Uludağ University]. Higher Council of Education.
- Bakır, E. E. (2023). *Blok tabanlı Scratch eğitimi ve uygulamalarının okul öncesi öğretmen adaylarının bilimsel yaratıcılık ve bilgi işlemsel düşünme becerilerine etkisinin incelenmesi* (Thesis number. 805462) [Master's Thesis, Kastamonu University]. Higher Council of Education.
- Büyüköztürk, Ş. (2023). *Sosyal bilimler için veri analizi el kitabı istatistik, araştırma deseni SPSS uygulamaları ve yorum*. Pegem Academy.
- Büyüköztürk, Ş., Kılıç-Çakmak, E., Akgün, Ö., Karadeniz, Ş. & Demirel, F. (2008). *Eğitimde bilimsel araştırma yöntemleri*. Pegem Academy. doi.org/10.14527/9789944919289
- Çelik, E. (2024). *Scratch kullanımının matematiksel problem çözme becerilerine etkisi*. (Thesis number. 868062) [Master's Thesis, Erciyes University]. Higher Council of Education.
- Erol, O. (2016). *Scratch ile programlama öğretiminin bilişim teknolojileri öğretmen adaylarının motivasyon ve başarılarına etkisi* (Thesis number. 395186) [Doctoral Thesis, Anadolu University]. Higher Council of Education.
- Erol, O. & Çırak, N. S. (2022). The effect of a programming tool scratch on the problem-solving skills of middle school students. *Education and Information Technologies*, 27(3), 4065-4086. <https://doi.org/10.1007/s10639-021-10776-w>
- Fagerlund, J., Häkkinen, P., Vesisenaho, M. & Viiri, J. (2021). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, 29(1), 12-28. <https://doi.org/10.1002/cae.22255>
- Gagné, R. M. (1985). *The conditions of learning and theory of instruction* (4 edition). Holt, Rinehart and Winston.
- Genç, Z. & Karakuş, S. (2013). Tasarımla öğrenme: eğitsel bilgisayar oyunları tasarımında Scratch kullanımı. In Z. Genç, *International Computer & Instructional Technologies Symposium* (pp. 22-24). 5th International Computer & Instructional Technologies Symposium. Elazığ, Türkiye.
- Green, A. J. & Gilhooly, K. (2005). *Problem solving*. In Cognitive psychology (pp. 347-366).

- Gültepe, A. (2018). Kodlama öğretimi yapan bilişim teknolojileri öğretmenleri gözüyle öğrenciler kodluyor. *Uluslararası Liderlik Eğitimi Dergisi*, 2(2), 50-60.
- Howard, N. R. & Howard, K. E. (2020). Coding+ Math: Strengthen K–5 math skills with computer science. *International Society for Technology in Education*.
- Hwang, W. Y., Wang, C. Y., Hwang, G. J., Huang, Y. M. & Huang, S. (2008). A web-based programming learning environment to support cognitive development. *Interacting with Computers*, 20(6), 524-534. <https://doi.org/10.1016/j.intcom.2008.07.002>
- Ioannidou, A., Bennett, V., Repenning, A., Koh, K. H. & Basawapatna, A. (2011). *Computational thinking patterns* [White paper]. AERA. <https://files.eric.ed.gov/fulltext/ED520742.pdf>
- Karataş, Z. (2015). Sosyal bilimlerde nitel araştırma yöntemleri. *Manevi Temelli Sosyal Hizmet Araştırmaları Dergisi*, 1(1), 62-80.
- Korkmaz, Ö. (2016). The effect of scratch-based game activities on students' attitudes, self-efficacy and academic achievement. *International Journal Modern Education and Computer Science*, 8(1), 16-23. <https://doi.org/10.5815/ijmecs.2016.01.03>
- Korkmaz, Ö. (2018). The effect of scratch-and Lego mindstorms Ev3-Based programming activities on academic achievement, problem-solving skills and logical-mathematical thinking skills of students. *MOJES: Malaysian Online Journal of Educational Sciences*, 4(3), 73-88.
- Lau, W. W. & Yuen, A. H. (2011). Modelling programming performance: Beyond the influence of learner characteristics. *Computers & Education*, 57(1), 1202-1213.
- Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51-61. <https://doi.org/10.1016/j.chb.2014.09.012>
- Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T. & Mendes, A. J. (2018). Learning computational thinking and scratch at distance. *Computers in Human Behavior*, 80, 470-477.
- Mayer, R. E. (1998). Cognitive, metacognitive, and motivational aspects of problem solving. *Instructional Science*, 26(1), 49-63.
- McGehee, J. J. (2001). Developing interdisciplinary units: A strategy based on problem solving. *School Science and Mathematics*, 101(7), 380-389.
- MoNE. (2023). *Matematik Dersi Öğretim Programı* [White paper]. MEB. <https://mufredat.meb.gov.tr/ProgramDetay.aspx?PID=329>
- Miller, M. & Nunn, G. D. (2001). Using group discussions to improve social problem-solving and learning. *Education*, 121(3), 470.
- OECD. (2013). “Problem-Solving Framework”, in *PISA 2012 Assessment and Analytical Framework: Mathematics, Reading, Science, Problem Solving and Financial Literacy* [White paper]. O. Publishing. <https://doi.org/10.1787/9789264190511-en>
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.
- Polya, G. (1945). *How to solve it: A new aspect of mathematical method* (Vol. 85). Princeton University Press.
- Repenning, A. (2006). *Excuse me, I need better AI! Employing Collaborative Diffusion to make Game AI Child's Play*. In A. Heirich, Sandbox '06: An ACM Video Game Symposium 2006 (pp. 169-178). Association for Computing Machinery. Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames. Boston, Massachusetts <https://doi.org/10.1145/1183316.1183341>
- Ross, A. & Willson, V. L. (2018). *Basic and advanced statistical tests: Writing results sections and creating tables and figures*. Springer.

- Schoenfeld, A. H. (1983). The wild, wild, wild, wild, wild world of problem solving (A review of sorts). *For the learning of mathematics*, 3(3), 40-47.
- Şeker, Z. (2020). Dinleme ve konuşma becerilerine yönelik lisansüstü tezlerin anahtar kelimeleri üzerine bir inceleme: Betimsel analiz. *RumeliDE Dil ve Edebiyat Araştırmaları Dergisi*(19), 128-140.
- Taylor, M. (2019). Class Notes: Algorithmic Thinking. 2023, Mayıs 2 tarihinde <https://www.cs.cmu.edu/~112-s23/notes/notes-algorithmic-thinking.html> adresinden erişilmiştir.
- Voskoglou, M. G. & Buckley, S. (2012). *Problem solving and computational thinking in a learning environment*. arXiv preprint arXiv:1212.0750. <https://doi.org/10.48550/arXiv.1212.0750>
- Werner, L., Campe, S. & Denner, J. (2012). *Children learning computer science concepts via Alice game-programming*. In L. S. King, Special Interest Group on Computer Science Education (SIGCSE) (pp. 427-432). Proceedings of the 43rd ACM Technical Symposium on Computer Science Education. Raleigh North Carolina, ABD. <https://doi.org/10.1145/2157136.2157263>
- Wing, J. M. (2010). *Computational Thinking: What and Why?* [White paper]. CMU. <https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf>
- Woo, K. & Falloon, G. (2022). Problem solved, but how? An exploratory study into students' problem solving processes in creative coding tasks. *Thinking Skills and Creativity*, 46, 101193. <https://doi.org/10.1016/j.tsc.2022.101193>
- Yalçinkaya, B., Dönmez, A., Aydın, F., Kayalı, N. & Sönmez, A. (2018). *İlköğretim çocuklarının kodlama algısı üzerine empirik bir analiz çalışması ve çocuk kütüphanelerinde uygulanmasının önemi*. In A. K. Yıldız, Uluslararası Çocuk Kütüphaneleri Sempozyumu Bildirileri (pp. 126-138). 1. Uluslararası Çocuk Kütüphaneleri Sempozyumu Bildirileri. Nevşehir, Türkiye.
- Yıldırım, A. & Şimşek, H. (2021). *Sosyal bilimlerde nitel araştırma yöntemleri*. Seçkin Publishing.
- Yılmaz, T., & İzmirli, S. (2023). Effect of unplugged and plugged coding activities on secondary school students' computational thinking skills. *Journal of Educational Technology and Online Learning*, 6(4), 1180-1193. <https://doi.org/10.31681/jetol.1375335>
- Zhang, L. & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education*, 141, 103607.