IJCSES

Volume 4, No: 3
January 2021
ISSN 2513-8359

International Journal of Computer Science Education In Schools

Editors

Dr Filiz Kalelioglu

Dr Yasemin Allsop

www.ijcses.org

International Journal of Computer Science Education in Schools

January 2021, Vol 4, No 3

DOI: 10.21585/ijcses.v4i3

Table of Contents

	Page
Umit Demir The Effect of Unplugged Coding Education for Special Education Students on Problem-Solving Skills	3 - 30
Breeann FLESCH[1], Camila GABALDÓN[1], Matthew NABITY[1], Darryl THOMAS[1] Choreographing Increased Understanding and Positive Attitudes towards Coding By Integrating Dance	31 - 48
Mahmut Can SÖZERİ[1], Serhat Bahadır KERT[2] Ineffectiveness of Online Interactive Video Content Developed for Programming Education[1]	49 - 69

The Effect of Unplugged Coding Education for Special Education Students on Problem-Solving Skills

Ümit DEMİR[1]

[1]Canakkale Onsekiz Mart University

DOI: 10.21585/ijcses.v4i3.95

Abstract

During recent years coding education has been an important issue in many countries. Coding education has been an important topic for these countries. One of the reasons why coding education is being discussed by educators and other partners of the schools is that it is seen as a key competence for students, and workers at developing problem-solving skills. Coding as an academic skill is seen as a part of logical reasoning. Coding is also accepted as one of the skills called "21st-century skills" required from individuals. Special education students are in a disadvantaged situation as in other learning platforms. Thus, this study aims to analyze the place of coding education in developing problem-solving skills of special education students. Within the scope of the study, unplugged coding applications were carried out with the participation of 34 students having mild intellectual disabilities who are continuing their education in a special education vocational school aged between 14 and 18. A question form was used to evaluate problem-solving skills. There was a significant difference between the pre-course and post-course skills of the students. Students' average scores of problem-solving skills in the post-course was higher than their average scores in the pre-course. The analysis of the findings showed that the students' skill scores in using problem-solving steps have increased in all these steps.

Keywords: special education, unplugged coding, problem-solving skill

1. Introduction

21st-century skills include basic skills areas such as critical thinking, problem-solving, and decision making, collaboration and communication, information literacy, technology literacy, flexibility and adaptability, global competencies, and financial literacy (Kereluik et al., 2013). The new 21st century skills are based on reasoning and logical thinking (Durak & Şahin, 2018). Because of this coding skills are important to get 21st-century skills. The ability of coding is called "algorithmic thinking" and "computational thinking" in different research (Durak & Şahin, 2018). In performing the coding process, it is important to follow the steps of comprehending, analyzing, solving the problems, and making the results as algorithms, establishing the correct algorithm, and encoding the algorithm with a program over the language. In this process, coders should use their algorithmic thinking or computational thinking skills (Lee et al., 2011).

In the economy, coding and programming skills have gained importance in many sectors and fields (Arora et al., 2001). The coding and programming skills included in the digital competencies with lifelong learning competence are among the important skills of the 21st century and the future (van Laar et al, 2017). The current workplace needs highly skilled employees that can work in increasingly interactive and tasks. Such employees are expected to effectively choose and use information from the number of accessible information and effectively use them in their works (Ahmad et al., 2013; Carnevale & Smith, 2013). For this reason, coding education should take place in education policies for the development of coding skills. Teachers have great responsibilities for the development of individuals with lifelong learning competencies (Durak & Şahin, 2018). Students can gain problem-solving, critical thinking, independent thinking, sharing, ethical behavior, knowledge, and digital literacy skills via programming and coding education. (Durak & Şahin, 2018; Voogt et al., 2013).

Programming education at an early age is becoming increasingly important. In this context, radical changes have been made in the educational curriculums around the world and in Turkey to provide programming education in elementary school (Demirer & Sak, 2016). In this regard, the European countries, and other developed countries as South Korea have begun to provide programming education to children and young people starting from primary school (Boccani et al, 2016; Demirer & Sak, 2016; Salter, 2013). The United Kingdom has integrated computer programming as a main course starting from primary school (Jones, 2013). The European Union (EU) organizes various events like European Code Week which is held in November. In the framework of the European Code Week, activities were held in various European countries, approximately three hundred workshops were organized, and thousands of students participated in these workshops (Demirer & Sak, 2016). In the programming education process, both

plugged and unplugged activities are used. While unplugged programming broadly refers to learning computational thinking and computer science concepts without relying on computational devices, plugged programming relies on the usage of computational devices in these learning processes (Aranda & Ferguson, 2018). Unplugged coding can include role-playing, manipulation of real-world objects (eg sticky notes, cards, wooden blocks), and physical actions of the body activity (Aranda & Ferguson, 2018). Special education students are also disadvantaged in this coding education as in other training. Most of the studies and applications for special education are prepared for gifted students (Alkan, 2019; Hagge, 2017; Lee, 2011). In these studies, plugged visual programming tools as Kodu and Scratch are used.

Empirical research on the impact of coding education on special education students is very limited (Adams & Cook, 2013; Miller, 2009; Topal, Budak & Geçer, 2017). Adams and Cook (2013) evaluated the effect of using programming and controlling robots' game for people with dementia about the stimulation of social behavior. As a result, it was found out that social behavior occurred more often than non-social behavior during the sessions. Miller (2009) reported that programming (Logo software) education improved both the programming skills and applying written vocabulary in a purposeful, rule-based manner. Topal, Budak, and Geçer (2017) found out that algorithm teaching via Scratch was effective on the problem-solving skills of deaf-hard hearing students. Coding activities attended by students with intellectual disabilities are also extremely limited (Karna-Lin et al., 2006; Ratcliff & Anderson, 2011; Taylor, 2018; Taylor, Vasquez, & Donehower, 2017; Wainer et. al., 2010). Karna-Lin et al. (2006) worked with children, 8 to 18 years old having learning difficulties and mild cognitive delays. Students used Lego robots in the experimental process. As a result, it was found out that students' social abilities as asking for advice, sharing ideas increased. Students also had an opportunity to practice their problem solving, logical thinking abilities. Ratchlif and Anderson (2011) reported that using programming education software (Logo) captured the students' interest. This programming experience was also described as a viable source of interactive challenge and problem-solving experience that provided a great deal of pride, intrinsic reward, enjoyment, and a sense of ownership of learning by the attending students. Taylor, Vasquez, and Donehower (2017) found out that despite programming education for students with mild disabilities was not effective in enhancing problem-solving abilities, programming in Logo captured students' interest. Taylor (2018) researched the potential effect of learning skills in computer programming for students with intellectual disabilities. Students were assessed through baseline, treatment, and generalization phases. As a result, students were successful at programming the robot. Wainer et. al. (2010) found out that interacting with Lego robots for programming education increased the collaborative behaviors of the students having Autism Spectrum Disorder (ASD). When these empirical studies were analyzed, it was observed that they generally include plugged applications. Coding training or practices involving only

plugged coding applications can only address students having computer usage skills. However, coding may provide many opportunities for special education students to support their learning and life-skills (Duff, McPherson, King & Kingsnorth, 2019; Taylor, Vasquez & Donehower, 2017). For this reason, the use of unplugged applications in coding training may enable a wider group of people with having different disabilities to benefit from this training. Consequently, it is expected that this study which investigates the development of special education students in using problem-solving steps with coding education will contribute to the literature. One of the main benefits of coding education is developing problem-solving skills (Chao, 2016; Hooshyar et al., 2016). Problem-solving skills can be improved by computational thinking processes. Through computational thinking, we can explain the problem and use simple methods or formulas to solve the problem by computer computation (Hsu et al., 2018).

1.1 Computational thinking

Computational thinking (CT) skills have very high importance in obtaining 21st-century learning skills (Barut et al., 2016). Wing (2008) defines computational thinking as "designing systems, solving problems, and understanding human behavior by drawing on the concepts main to computer science". She stated that computational thinking includes some common concepts, such as data representation, problem decomposition, and modeling. She also stated that "computational thinking is a basic skill for everyone, not just for computer scientists. We should add computational thinking to every child's analytical ability like writing, reading, and arithmetic abilities." Thinking like a computer scientist enables children to solve any problem they can face logically. Computational thinking can help in designing solutions to automation-sensitive problems (Allsop, 2019). In addition to thinking like a computer scientist, there seems to be value in teaching critical thinking skills and the problem-solving mentality that comes with traditional writing, reading, and arithmetic. (Kazakoff, 2014).

Computational thinking includes problem-solving processes:

• Formulating problems in a way that let us use a computer and other tools to help solve these problems,

• Analyzing and organizing data logically,

• Representing data through abstractions, such as simulations and models,

• Automating solutions through algorithmic thinking (a series of ordered steps),

• Identifying, analyzing, and implementing possible solutions to ensure that the steps and resources are combined most efficiently and effectively,

• Generalizing and transferring this problem-solving process to various problems (Barr et al., 2011). As a result, computational thinking helps in the development of problem-solving strategies.

Problem-solving strategies as computational thinking and algorithmic thinking give us loads of talents and skills, including:

• Confidence in dealing with complexity,
• Instance on working with difficult problems,
• Tolerance for uncertainty,
• The ability to deal with open-ended problems,
• The ability to communicate and work with others to achieve a common goal or solution (Barr & Stephenson, 2011). Therefore, including problem-solving skills developing activities via computational thinking in the education process can provide loads of talents and skills that are of great importance to the students. In the process of realizing these activities, evaluating, and following the development levels of students is of great importance in terms of structuring the teaching processes. The Computational skill assessment model developed by Allsop (2019) includes three aspects: 'computational concepts', 'metacognitive practices', and 'learning behaviors'. Computer game design processes are generally used in evaluating these dimensions (Allsop, 2019; Brennan & Resnick, 2012; Werner et al., 2012; Werner et al., 2014).

1.2 Algorithmic Thinking

Some difficulties in mathematics are generally associated with weaknesses in algorithmic problem-solving (Plerou & Vlamos, 2016). The algorithm also describes a finite sequence of actions that describe how to solve a given problem (Kotthoff, 2016). The ability to design and use algorithmic forms or schemas in problem-solving processes pre-requires cognitive skills and enforces these skills (Antonia, Panagiotis & Panagiotis, 2014). This means that generally algorithmic techniques can be applied similarly to solve a problem (Atmatzidou & Demetriadis, 2016). Algorithmic thinking is a critical skill in getting problem-solving skills. Algorithmic thinking is defined as the ability to construct new algorithms to solve given problems (Futschek, 2006). In algorithmic thinking, a set of solution rules including devising a step-by-step solution are designed and application instructions for rules are defined by steps (Angeli et al, 2016; Curzon et al., 2014). Algorithmic thinking includes understanding, applying, evaluating, and producing algorithms skills (Kanbul & Uzunboylu, 2017). In these activities putting actions in the correct sequence and using flow control are very important elements. These are also considered important for computational thinking (Angeli et al, 2016).

In the research in which the effect of using algorithmic thinking in programming activities on the development of computational and mathematical thinking was examined, it was found out that coding and programming education is effective for developing problem-solving strategies, teaching mathematics, and for creative thinking. (Taylor et al., 2010; Kalelioglu & Gulbahar, 2014). Algorithmic thinking is not only an important part of computer science, but it is also an

important part of our lives especially in decision-making (Kátai, 2015; Mohaghegh & McCauley, 2016). Decision-making processes are very important for individuals with intellectual disabilities. They need supports in decision-making processes (Jamesson et. al., 2015). As a result, algorithmic thinking can be an important supporter and facilitator of problem-solving and decision-making processes.

1.3 Problem-Solving

Problem-solving is a critical learning process in formal education activities for all the education levels from primary to higher education (Jonassen, Howland, Moore, & Marra, 2003; Lazakidou & Retalis, 2010). This advanced cognitive ability can be understood as the ability to use rules and concepts to solve the problem (Wang, Han, Zhan, Xu, Liu & Ren, 2015). Educators expect that their students will graduate from their courses with good problem-solving skills. They use open-ended problems and activities to develop these skills (Woods et al., 1997). Students' participation in problem-solving activities assists them to gain helpful attitudes that are crucial to real life, such as creativity, flexibility, thinking, and efficiency. Besides, all these attitudes have already been linked to life-long learning skills (Goffin & Tull, 1985). So, problems used in the educational process should be real-life scenarios that provide students opportunities to become real-life problem solvers (Yu, Fan, & Lin, 2015).

To date, many problem-solving models have been suggested (Bransford & Stein, 1993; Good & Brophy, 1995; Hohn & Frey, 2002; Ormrod, 2000; Polya, 1973; Sternberg, 2003). A well-known problem-solving model is Sternberg's 7 steps model. Sternberg's 7 steps model can be used to both well-defined (known) and ill-defined problems in which learners engage a different set of epistemic beliefs (Jonassen, 2010; Lazakidou & Retalis, 2010). This model includes the following steps: problem identification, the definition of the problem, constructing a strategy, organizing information, allocation of resources, monitoring, and evaluating problem-solving. (Sternberg, 2003: 360). These steps of the problem-solving process require one to arrange each step and make decisions at the same time (Özsoy & Ataman, 2017). Also, the learner should experience and make strong the problem-solving process continues to complete the task (Wang, Han, Zhan, Xu, Liu & Ren, 2015).

Problem-solving skills have a potential impact on the individuals' having intellectual disabilities independence and academic achievement (Erickson, Noonan, Zheng, & Brussow, 2015; Root, Saunders, Spooner, & Brosh, 2017). By using schema-based instruction in the practice process of teaching problem solving to students with mild intellectual disabilities, these students can benefit from many instructional features (Jidentra et al., 2015). Because of this visual representation are very important for special education students (Root, Saunders, Spooner, &

Brosh, 2017). In problem-solving processes, some thinking strategies are used. More common of these are algorithmic thinking and computational thinking.

1.4 Research Questions

During recent years coding education has been an important issue in many countries to support and develop problem-solving skills. One of the reasons why coding education is being discussed by educators and other partners of the schools is that it is seen as a key competence for students and workers. Coding as an academic skill is seen as a part of logical reasoning. It is also accepted as one of the skills called "21st-century skills" required from individuals (van Laar et al, 2017). Learning how to code is equally valuable as learning math, reading, and writing (Horizon, 2015). It is very important to include coding educations with plugged or unplugged activities in the training process. It is thought that especially unplugged coding activities will appeal to a much wider group of students with having different disabilities. Lechelt et al. (2018), in their research, found out that a physical toolkit (magic cube) could be used both supporting comprehensions of computational concepts and enabling students to get excited about learning with fun. Because of these, computer-aided coding activities may only address students having computer usage skills. As a result, this study aims to analyze the effect of coding education having unplugged activities in developing problem-solving skills of special education students. Within the aim of this research following questions are identified:

RQ1: What is the effect of unplugged coding education on the problem-solving skills of special education students?
RQ2: What is the effect of unplugged coding education on the steps of the problem-solving skill of special education students?

2. The study

2.1.Method

An experimental method was used in the study. Among the different types of these methods, the full experimental method is the method with the highest scientific value. Thus, the full experimental method is used in this research. Within the scope of this study, both experimental and control groups were formed according to unplugged coding education materials usage status.

2.2.Participants

This study was conducted with 34 students having mild intellectual disabilities attending a Math education in the Çanakkale Special Education Vocational and Technical High School, Turkey. Students with a mild intellectual disability typically defined by having an IQ between 55 and

70 and having impairments in adaptive skills, such as daily living, social skills, and communication (Schalock, et al., 2010). The distribution of the students according to demographic characteristics is shown in Table 1. The number of females (n=14) and males (n=11) students are close to each other and the average age of the students is approximately 15.5. In the sample choosing process, students' parental written permission and their voluntary participation were asked. All the students wanted to participate in the experimental process.

Table 1. Distribution of demographic features of students (n=25)

Gender	**f**	**%**
Female	14	56
Male	11	44
Age	**f**	**%**
14	5	20
15	7	28
16	6	24
17	5	20
18	2	8
Total	25	100

2.3.Methods of Data Collection

The full experimental design was used in this research. There are 4-6 students in the classes in the special education school. At each class level, this school has two classes. The experimental processes were carried out in the mathematics course. Before the experimental processes, each class level (9th, 10th, 11th) categorized randomly as one control and one experimental group. The data collection process consists of 4 basic stages (Figure 1).

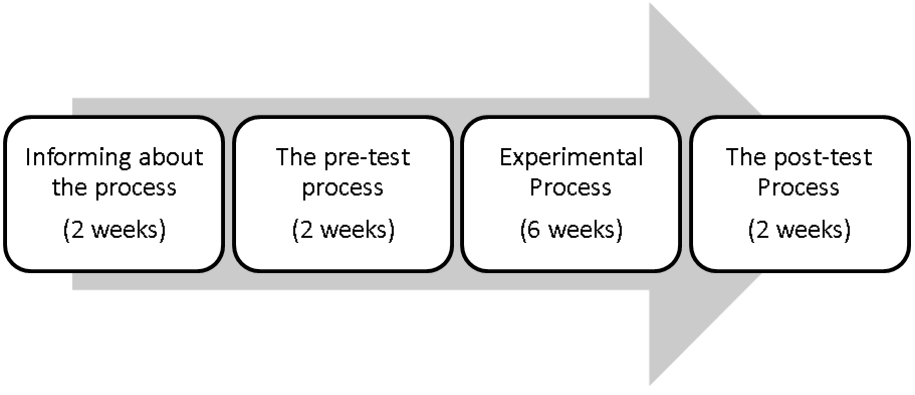


Figure 1. The data collection process steps

2.3.1. Informing About the Process

Before the experimental process, the math teacher was informed about the aim and process of the research including problem-solving steps, materials that will be used by students, and what to pay attention to in the evaluation process.

A performance rating form for determining the status of using problem-solving skills was administered to the students in terms of their skills in using problem-solving. In this form, including checklist items, students' usage of problem-solving steps defined by Sternberg (2003) are questioned. The problem-solving cycle's steps are problem identification, the definition of the problem, constructing a strategy, organizing information, allocation of resources, monitoring, and evaluating the problem-solving stage. The first step "problem identification" was not included in this research. Because the problem situation was given to the student and asked to find a solution to the problem. In the problem identification step, students are wanted to answer the question of "Do we actually have a problem?" (Sternberg & Sternberg, 2012:445). In pre-test and post-test processes, the teacher was wanted to ask the students about each problem-solving stage defined in the question form (Appendix A). In filling the performance rating form, students' written answers to the questions, and their verbal explanations of these written answers on problem-solving steps were evaluated. A blank sample of the student question form (translated to English) is presented in appendix A. In preparing both the question form asked to the students and performance rating form Sternberg & Sternberg (2012:445-446) and Lazakidou & Retalis (2010:5) were used. In the question form, students wanted to identify the problem, define the problem, construct a strategy, organize the information, allocate the resources, and monitor and evaluate their solving strategy. In the performance rating form, students' verbal and written answers to the questions were evaluated. In the performance rating form, problem-solving skills are scored out of 5 for each problem-solving cycle step (1: insufficient, 2: acceptable, 3: intermediate, 4: good, 5: very good). In filling this form, the math teacher assisted in evaluating the verbal and written answers of the special education students because of the communication and writing problems of some students. The main aim of the study was not to give special education students the skill of sorting the algorithm problems correctly. The main aim was to develop special education students' problem-solving skills in a logical framework by using problem-solving steps. For example, the student may find the correct sorting in the pre-test by chance or by memorizing in the post-test. But for the research, without creating a logical framework by using problem-solving steps sorting the algorithm was not valuable. At the beginning of the experimental process, the experimental group students also were informed about the rules of the games and problem-solving steps. After the informing process, the experimental process was started.

2.3.2 The Pre-test Processes

In the pre-test and post-test process (applied following 6 weeks), algorithm cards shown in figure 2-3-4 and the question form (Appendix A) were used. Both the students in the control and experimental groups were asked to solve "Buying Bread", "Washing Dishes" and "Making Pasta" problems by using problem-solving steps. At the same time, students were asked to solve these problems by answering the questions on the student question form. In the pre-test students' written answers to the questions and their verbal explanations of these written answers on problem-solving steps were evaluated by the performance rating form. The opinions of the course teacher about student performance were also taken into consideration during the evaluation process.

EKMEK ALMAK	**BUYING BREAD**
Sokağa çık	……. Get out on the street
Dükkana gir	……. Enter the shop
Kasaya götür	……. Take the product to the payment case
Parayı öde	……. Pay the money
Eve dön ve sofraya koy	……. Go back home and put it on the table

Figure 2. Buying bread and 5 algorithmic steps (Turkish –English)

In this process, the teacher was wanted to ask the students about each problem-solving stage defined in the question form. In figure 2, buying bread step cards (including: get out on the street, enter the shop, take the product to the payment case, pay the money, go back home, and put it to the table) are shown. In figure 3, washing dishes step cards (including cleaning the kitchen midden, rinsing out, soaping, rinsing, drying) are shown. In figure 4, making pasta cards (including boiling, opening the package, putting it into the water, waiting for 15 minutes, pouring to the strainer) is shown.

BULAŞIK YIKAMAK	**WASHING DISHES**
Artıkları sıyır	Cleaning the kitchen-midden
Sudan geçir	Rinsing out
Sabunla	Soaping
Durula	Rinsing
Kurut	Drying

Figure 3. Washing dishes (title) and 5 algorithmic steps (Turkish –English)

In each game, firstly students were wanted to define the problem with their own words. Then they were wanted to construct a strategy to solve the defined problem and explain the choosing reason for that strategy. After this step, they were wanted to organize having information and explain how to use these in finding a solution. After this problem-solving step, they were wanted to allocate resources (time, effort, etc.) in solving the problem. Following this step, they were wanted to question themselves if they are on true track as they proceed to solve the problem. At the last step, they were wanted to evaluate their problem-solving strategy if the strategy worked or not. If the discovered strategy did not work, they were wanted to explain the reasons. In all problem-solving steps, all students were wanted to write their problem-solving steps and make an oral presentation of their solving. These written answers were evaluated by the researcher and the teacher.

MAKARNA YAPMAK	**MAKING PASTA**
Suyu kaynat	Boiling
Paketi aç	Opening the package
Suyun içine at	Putting into water
15 dk bekle	Waiting for 15 minutes
Süzgece dök	Pouring to the strainer

Figure 4. Making pasta (title) and 5 algorithmic steps (Turkish –English)

2.3.3 The Pre-test Processes

In the classroom environment, the experimental group students played three games to learn the usage of problem-solving steps to solve problems in the teaching process. They played the games individually and as a member of the game group under the math teacher supervision. In the selection of the games, expert opinions were received from the special education teachers working in the special education school where the application was carried out and the academic staff of Çanakkale Onsekiz Mart University Special Education department. The first game was "The wolf, the lamb, the weed and the farmer activity" (figure 5). This game includes real-life concepts that students can easily fictionalize. Concrete and close-to-life learning environments are crucial for the success of the teaching process (Holt, Segrave & Cybulski, 2013).

Figure 5. The wolf, the lamb, the weed, and the farmer activity

In this game, Uncle Ahmet's farm was just outside of the village, just across the river. Uncle Ahmet took his lamb one day, the wolf descending from the forest to his garden, and a certain amount of grass he had reserved for his lamb and wanted to cross over to the shore. But the only way he could cross over was a small boat and it was impossible to cross them all together. He could take one to her at a time; He can either put the lamb, the wolf, or the weed. If the farmer leaves the lamb and wolf, the wolf eats the lamb. If the farmer leaves the lamb and the weed, the lamb eats the weed. So, the question of the game is "How do you think Uncle Ahmet will get all three of them across? The solution of this game consists of 5 steps. It is important for special education students facing problems to have fewer solution steps especially at the beginning of the learning process. Because students with disabilities rely on considerable step by step instructions (Mechling & Ortega-Humdon, 2007). In this game, problem-solving steps are:

(1) Transporting the lamb across the river,
(2) Transporting the wolf across the river,
(3) Transporting the lamb back,
(4) Transporting the weed across the river.
(5) Transporting the lamb across the river.

The second game played by students was “Tower of Hanoi” (Figure 6). The Towers of Hanoi is a puzzle that has been studied by computer scientists and mathematicians for many years. The goal is to recreate the 4-disk tower on the third post. The monks must move the disks according to two rules:
(1) The monks can only move one disk at a time.
(2) The monks can only place smaller disks on top of larger disks.

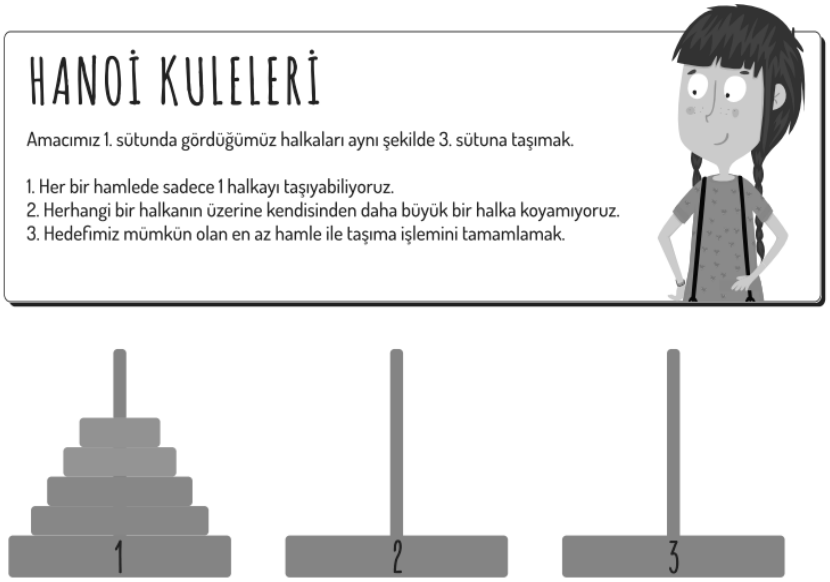


Figure 6. The towers of Hanoi activity

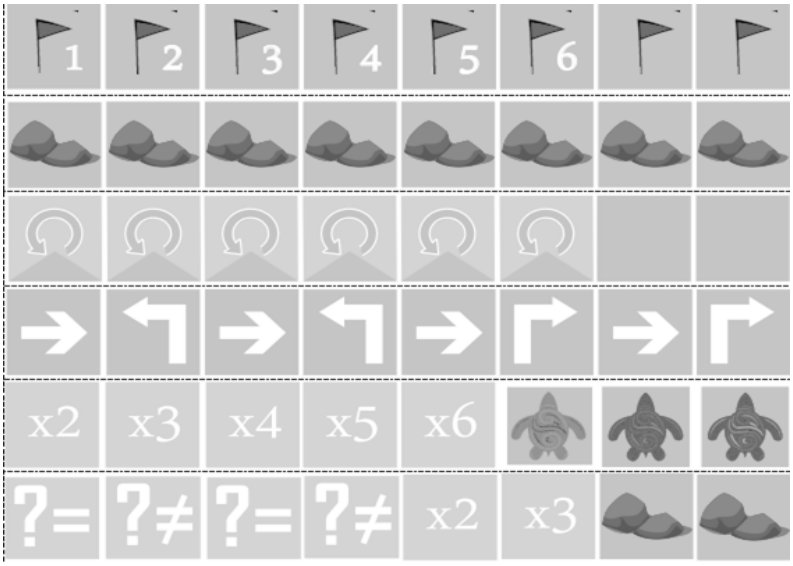


Figure 7. Tospaa unplugged coding game algorithm cards

The third game played by students was the "Tospaa Unplugged Coding Game". The game aims to bring the turtle to the targets without getting stuck. To reach the targets, algorithm cards in figure 7 are used. In this game, one of the most important terms of programming, the loop concept can be taught easily with this game. As a group game, gamers make algorithms to reach their targets. The gamer that uses the least algorithm card wins the game. An example scenario of the Tospaa game is shown in figure 8. At the same time, the control group students were informed about the problem-solving steps by the researcher. In this process, firstly the same math teacher gave the basic information about "why we need to use problem-solving steps?" and the problem-solving steps. In this process, the questions needed to answer in each problem-solving step were explained in detail by the researcher. Then different problems were asked students with worksheets. Students' answers for problem-solving steps were discussed in teacher management. Students were informed about their mistakes.

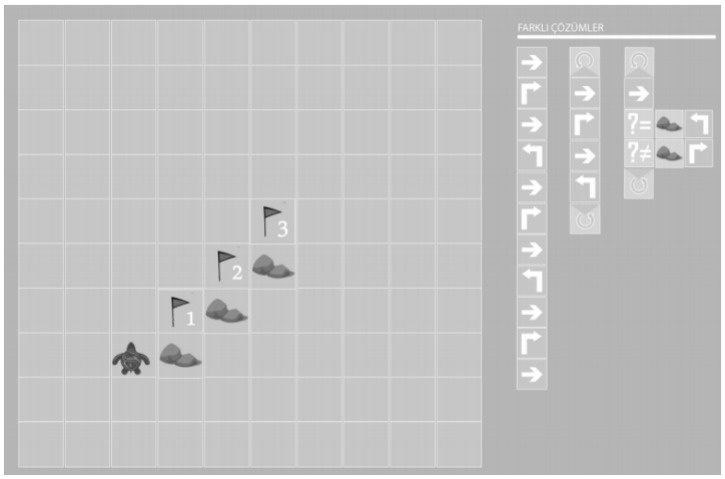


Figure 8. An example scenario of tospaa unplugged coding game

2.4.Data Analysis

Data collection involved by the performance rating form for determining the status of using problem-solving skills was administered to the students in terms of their skills in using problem-solving. The pre-test and post-test scores obtained by the students on this form were used in data analysis. These analysis values are given in the results section.

3. Results

3.1 Question 1: What is the effect of unplugged coding education on the problem-solving skills of special education students?

The pre-course skills of students were determined before the application while their post-course skills were determined just after the application. The results regarding whether a significant difference occurred between the scores in terms of skill change are presented in Table 2. Table 2 demonstrates that there is a significant difference [$t_{(24)}$=-7.19, p<.001] between the pre and post-course skills of the student group. While the average point of the pre-course skill was 10.68, the post-course average point increased to 13.36.

Table 2. Pre-course and post-course comparison of skill scores (n=25)

	N	$\overline{X}$	**Sd**	**t**	**p**
Pre-course skill scores	25	10.68	3.56	-7.19	.000**
Post-course skill scores	25	13.36	4.58		

*p< .01, ** p< .001

3.2 Question 2: What is the effect of unplugged coding education on the steps of the problem-solving skill of special education students?

Problem-solving skills were categorized into 6 steps. The results regarding whether a significant difference occurred between the scores in terms of skill change are presented in Table 3.

Table 3. Pre- and post-course skill comparison of the students (n=25)

Problem-Solving Step Name	Score Type	N	$\overline{X}$	Sd	t	p
Definition of the problem	Pre-course skill score	25	2.40	1.08	-4.10	.000**
	Post-course skill score	25	2.88	1.01		
Constructing a strategy	Pre-course skill score	25	2.48	1.05	-3.36	.003*
	Post-course skill score	25	2.80	1.04		
Organizing information	Pre-course skill score	25	2.12	1.01	-2.87	.008*
	Post-course skill score	25	2.44	.96		
Allocation of resources	Pre-course skill score	25	1.36	.49	-6.20	.000**
	Post-course skill score	25	2.16	.85		
Monitoring the process	Pre-course skill score	25	1.20	.41	-3.98	.001*
	Post-course skill score	25	1.72	.84		
Evaluating the process	Pre-course skill score	25	1.12	.33	-2.30	.031
	Post-course skill score	25	1.36	.57		

*p< .01, ** p< .001

Table 3 demonstrates that there are significant differences between pre and post-course observation scores of the student group in all the problem-solving skill steps (step 1: [$t_{(24)}$=-4.10, p<.001], step 2: [$t_{(24)}$=-3.36, p<.01], step 3: [$t_{(24)}$=-2.87, p<.01], step 4: [$t_{(24)}$=-6.20, p<.001], step 5: [$t_{(24)}$=-3.98, p<.01], step 6: [$t_{(24)}$=-2.30, p<.05]).

4. Discussion of Findings

This study evaluated the effect of the unplugged algorithm training for special education students. The study demonstrated that using unplugged algorithm training games can considerably improve problem-solving skills within all six steps. When the change in the problem-solving skill steps was examined, it was found that the most increase occurred in step 5 (allocating of resources) and the least increase was in step 7 (evaluating problem-solving stage). In step 5, students' experience of organizing information can be increased more with the experience they have with unplugged coding games. In steps 6 and 7, the monitoring and evaluating the problem-solving stage may require long-term development. Therefore, this step may require longer-term practice and experience. As the pre-test scores in the first 3 steps are not too low, the increase may be limited. As a result, using unplugged algorithm games improve the special education students' problem-solving skill. This result is important to see the importance of giving coding education.

The effect results of algorithm education consisted of the research results conducted by Alotaibi; Allan & Kolesar (1996), Erdem (2018), Fessakis et al. (2013), Howland & Good (2015); Topal, Budak, and Geçer (2017). Allan & Kolear (2011) gave algorithm and coding education in the computer science course (CS1) at Utah State University. In the course, students gained mathematical and problem-solving skills while becoming familiar with the computer as a tool and learning. Erdem (2018) found out that coding and algorithm education developed 5th-grade students' problem-solving skills. Fessakis et al. (2013) found out in their research that 5–6 years old kindergarten children were pleased with the attractive learning activities and had chances to improve mathematical concepts, social and problem-solving skills. Howland and Good (2015), there were significant improvements in 12–13-year-olds teenagers' computational communication and problem–solving skills after using algorithm developing applications for creating games. Topal, Budak, and Geçer (2017) found out that algorithm teaching via Scratch was effective on the problem-solving skills of deaf-hard hearing students. There are also research results pointing out that algorithmic thinking is not effective in learning and problem-solving (Doleck, Bazelais, Saxena & Basnet, 2017; Psycharis & Kallia, 2017).
Doleck, Bazelais, Saxena & Basnet (2017) found out a lack of association between computational thinking skills and academic performance. They emphasized that this result maybe since the curriculum has yet to be adequately associated with 21st-century skills teaching. Pyscharis & Kallia (2017) found out in their research conducted with 66 high school students that programming education including algorithmic procedures enhanced students' reasoning skills but did not enhance their problem-solving skills. As a result, When the studies using algorithmic structures in coding education are examined it is seen that it generally affects

problem-solving skills positively. It is especially important to make the curriculum supportive of problem-solving skills within 21 st-century skills teaching (Doleck, Bazelais, Saxena & Basnet, 2017; Israel, Wherfel, Pearson, Shehab & Tapia, 2015; Psycharis & Kallia, 2017). There are several instructional benefits for students that can get from the inclusion of problem-solving skills within K-12 programs. They can benefit from this skill in building higher order thinking skills and increasing collaborative problem-solving (Kafai & Burke, 2014). For this reason, it is of great importance to add the skills for the acquisition of 21st-century skills to the curriculum and to evaluate them.

In Turkey, it is observed that there is not still sufficient importance given to robotic applications and coding education within 21st-century skills. As a result of institutional studies by universities, there are also effective robotic studies in almost all universities. However, there is not enough initiative to integrate coding education into the programs of different education levels from pre-school to university. But robotic coding education also can create lots of benefits for students. It can facilitate advanced hands-on programming, increase the rate of two-directional communication between the students and the robot (Virnes, Sutinen & Kärnä-Lin, 2008). The course of information technologies and software development (ITSD) course in the education program of Turkey is included as an elective course in 5th, 6th, 7th, and 8th grades based on the curriculum published in 2012 (Kanbul & Uzunboylu, 2017). In the last curriculum of ITSD course developed in 2017, not only setting integrity between informatics technology units but also with other subjects and real life is important. so that it is important to realize the discourses in real life (students benefit from informatics technologies and software development courses) and to learn how to use technology appropriately (Karaman & Karaman, 2019). We cannot say that this change in planning is sufficient in practice. Coding training is carried out under the supervision of the IT teachers and in their use with a single interactive board. It is considered that it is important to provide coding laboratory facilities for student-centered practice to realize effective learning. The number of coding classes that have the infrastructure installed is quite limited. So, it would be wrong to expect to achieve success in coding with just the program change. Besides, secondary schools can be considered a bit too late for coding training.

Teaching coding to children in early childhood before elementary school education may enable long-term economic payoffs. Investments in early childhood interventions are associated with lower costs and longer-term impacts than later interventions in childhood. (Heckman, 2006; Heckman & Masterov, 2007; Reynolds, et al., 2011).

5. Conclusion

Besides, special education students are the most disadvantaged students in this education process. Measures can be taken to ensure that all disadvantaged students receive coding training. For example, Al-Khalifa & AlSaeed (2020) found out that tactile teaching strategies were effective in the programming education of students with vision impairment. In this research, it was also found out that students' problem-solving skills are insufficient. But problem-solving skill is an important skill for the development of life skills (Prajapati, Sharma & Sharma, 2017; Wurdinger & Qureshi, 2015). Similarly, special education students need to develop life skills to maintain their daily lives (Smith, Cihak, Kim, McMahon & Wright, 2017). For this reason, it is thought that education and practices that support problem-solving should be given importance. Based on the research results; it can be indicated that coding education can provide many educational opportunities to support the problem-solving skills of special education students. Some of these opportunities are the ability to break down problems into smaller parts and to draw on both logic and creativity to figure out the best ways to solve them (Lechelt et al., 2018). To inform and educate the future generation, companies, universities should make investments and serious ventures for coding and robotic applications education. Soon, new professions will emerge, and many occupations will not be needed. Therefore, it is very important to teach 21st-century skills to all children including students who need special education. It is thought that it is of great importance to increase the competencies and expertise of teachers in this field to develop students' problem-solving skills. In this context, it is thought that it would be beneficial to provide teachers with training or in-service training on how to use the applications for the development of problem-solving skills and how to evaluate the dimensions of problem-solving skills of the students. According to the findings of the study, the following suggestions were made.

• Coding education applications and web-based platforms like scratch, code.org should be integrated into the K12 curriculum.

• In-service or training courses should be given to teachers and teacher candidates for problem-solving steps, development, and evaluation of problem-solving skills.

• Coding laboratories including robotic and unplugged application materials should be set.

• Experimental studies that investigate the traditional programming course and unplugged programming/robotic programming course can be made.

Limitations

The research was carried out in a vocational high school where mild intelligent students have been given special education. Studies conducted with different samples of intellectual disability can be made. Also, in this research, unplugged coding education materials were used.

Comparative studies can be conducted on the effectiveness of computerized and unplugged coding education.

References

Adams, K. D., & Cook, A. M. (2013). Programming and controlling robots using scanning on a speech generating communication device: A case study. *Technology and Disability*, *25*(4), 275-286. doi:10.3233/TAD-140397

Ahmad, M., Karim, A. A., Din, R., & Albakri, I. S. M. A. (2013). Assessing ICT competencies among postgraduate students based on the 21st century ICT competency model. *Asian Social Science*, *9*(16), 32. doi:10.5539/ass.v9n16p32

Alkan, A. (2019). The effect of code game lab software on the level of problem-solving skills in programming language teaching. M*ehmet Akif Ersoy University Faculty of Education Journal*, (50), 480-493.doi:10.21764/maeuefd.486061

Allan, V. H., & Kolesar, M. V. (1996). Teaching Computer Science: A Problem-Solving Approach that Works. National Educational Computing Conference 1996, Minneapolis, MN. Retrieved from https://files.eric.ed.gov/fulltext/ED398878.pdf

Allsop, Y. (2019). Assessing computational thinking process using a multiple evaluation approach. *International journal of child-computer interaction*, *19*, 30-55. doi: 10.1016/j.ijcci.2018.10.004

Alotaibi, H., S Al-Khalifa, H., & AlSaeed, D. (2020). Teaching Programming to Students with Vision Impairment: Impact of Tactile Teaching Strategies on Student's Achievements and Perceptions. *Sustainability*, *12*(13), 5320. doi: 10.3390/su12135320

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. Journal of *Educational Technology & Society*, *19*(3), 47-57. Retrieved from https://www.jstor.org/stable/pdf/jeductechsoci.19.3.47.pdf

Antonia, P., Panagiotis, V., & Panagiotis, K. (2014). Screening Dyscalculia and Algorithmic Thinking Difficulties "1st International Conference on New Developments in Science and Technology Education" Proceedings Manuscripts.

Aranda, G., & Ferguson, J. P. (2018). Unplugged Programming: The future of teaching computational thinking? *Pedagogika*, *68*(3), 279-292. Retrieved from https://pages.pedf.cuni.cz/pedagogika/? attachment_id=11945&edmc=11945

Arora, A., Arunachalam, V. S., Asundi, J., & Fernandes, R. (2001). The Indian software services industry. *Research policy*, *30*(8), 1267-1287. Retrieved from

http://www1.ximb.ac.in/users/fac/dpdash/dpdash.nsf/23e5e39594c064ee852564ae004fa010/fc16012dc5a4d1cae52568b200183115/%24FILE/Soft_industry1.pdf

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, *75*, 661-670.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *Acm Inroads*, *2*(1), 48-54.doi: 10.1145/1929887.1929905

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, *38*(6), 20-23. Retrieved from https://files.eric.ed.gov/fulltext/EJ918910.pdf

Barut, E., Tugtekin, U., & Kuzu, A. (2016). An Overview of Computational Thinking Skills with Robotic Applications. *The 3rd International Conference on New Trends in Education (ICNTE)*. Retrieved from http://journals.sfu.ca/onlinejour/index.php/i-jet/article/download/6097/4264

Bransford, J. D., & Stein, B. S. (1993). *The ideal problem solver: A guide for improving thinking, learning and creativity (2nd ed.)*. NY: W.H. Freeman.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *In Proceedings of the 2012 annual meeting of the American educational research association*, Vancouver, Canada (Vol. 1, p. 1-25). Retrieved from http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Carnevale, A. P., & Smith, N. (2013). Workplace basics: The skills employees need and employers want. *Human Resource Development International*, *16* (5), 491-501. Retrieved from https://cew.georgetown.edu/wp-content/uploads/2014/11/HRDI.Editorial.pdf

Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*, *95*, 202-215. doi: 10.1016/j.compedu.2016.01.010

Curzon, P., Dorling, M., Ng, T., Selby, C., & Woollard, J. (2014). *Developing computational thinking in the classroom: a framework* Swindon, GB. Retrieved from https://eprints.soton.ac.uk/369594/1/DevelopingComputationalThinkingInTheClassroomaFramework.pdf

Demirer, V., & Sak, N. (2016). Programming education and new approaches around the world and in Turkey. *Journal of Theory and Practice in Education*, *12*(3), 521-546. Retrieved from

http://acikerisim.lib.comu.edu.tr:8080/xmlui/bitstream/handle/COMU/1443/Veysel_Demirer_Makale.pdf?sequence=1&isAllowed=y

Doleck, T., Bazelais, P., Lemay, D. J., Saxena, A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education*, *4*(4), 355-369. doi: 10.1007/s40692-017-0090-9

Duff, C., McPherson, A. C., King, G., & Kingsnorth, S. (2019). Deconstructing residential immersive life skills programming through a pedagogical lens: mechanisms that can facilitate learning for youth with disabilities. *Journal of Research in Special Educational Needs*. doi: 10.1111/1471-3802.12470

Durak, H. Y., & Şahin, Z. (2018). Investigation of the contribution of coding training in teaching candidates to the development of lifelong learning competencies. *Journal of Ege Education Technologies*, *2*(2), 55-67. Retrieved from http://dergipark.gov.tr/download/article-file/618107

Erdem, E. (2018). *The investigation of different teaching strategies during teaching programming process in block based environment in terms of different factors.* (Master's thesis, Başkent University Institute of Educational Sciences).

Erickson, A. S. G., Noonan, P. M., Zheng, C., & Brussow, J. A. (2015). The relationship between self-determination and academic achievement for adolescents with intellectual disabilities. *Research in Developmental Disabilities*, *36*, 45-54. doi: 10.1016/j.ridd.2014.09.008

European Commission (2018). Coding - the 21st-century skill. Retrieved from https://ec.europa.eu/ digital-single-market/coding-21st-century-skill

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, *63*, 87-97. doi:10.1016/j.compedu.2012.11.016

Good, T. L., & Brophy, J. (1995). *Contemporary educational psychology (5th ed.)*. NY: Longman Publishers.

Goffin, S., & Tull, C. (1985). Problem solving: Encouraging active learning. *Young Children, 40*(1), 28–32. Retrieved from https://psycnet.apa.org/record/1985-27772-001

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

Heckman, J. J. (2006). Skill formation and the economics of investing in disadvantaged children. *Science*, *312*(5782), 1900-1902. doi:10.1126/science.1130121

Heckman, J. J., & Masterov, D. V. (2007). The productivity argument for investing in young children. *Applied Economic Perspectives and Policy*, *29*(3), 446-493. Retrieved from https://www.nber.org/papers/w13016.pdf

Hohn, R., & Frey, B. (2002). Heuristic training and performance in elementary mathematical problem solving. *The Journal of Educational Research, 95*(6), 374–380. doi:10.1080/00220670209596612

Holt, D., Segrave, S., & Cybulski, J. L. (2013). E-Simulations for educating the professions in blended learning environments. In *IT Policy and Ethics: Concepts, Methodologies, Tools, and Applications* (pp. 1102-1123). IGI Global. doi: 10.3968/j.sll.1923156320110303.1200

Hooshyar, D., Ahmad, R. B., Yousefi, M., Fathi, M., Horng, S. J., & Lim, H. (2016). Applying an online game-based formative assessment in a flowchart-based intelligent tutoring system for improving problem-solving skills. *Computers & Education*, *94*, 18-36. doi: 10.1016/j.compedu.2015.10.013

Howland, K., & Good, J. (2015). Learning to communicate computationally with Flip: A bi-modal programming language for game creation. *Computers & Education*, *80*, 224-240. doi: 10.1016/j.compedu.2014.08.014

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*, 296-310. doi: 10.1016/j.compedu.2018.07.004

Israel, M., Wherfel, Q. M., Pearson, J., Shehab, S., & Tapia, T. (2015). Empowering K–12 students with disabilities to learn computational thinking and computer programming. *TEACHING Exceptional Children*, *48*(1), 45-53. doi: 10.1177/0040059915594790

Jameson, J. M., Riesen, T., Polychronis, S., Trader, B., Mizner, S., Martinis, J., & Hoyle, D. (2015). Guardianship and the potential of supported decision making with individuals with disabilities. *Research and Practice for Persons with Severe Disabilities*, *40*(1), 36-51. doi: 10.1177/1540796915586189

Jitendra, A. K., Petersen-Brown, S., Lein, A. E., Zaslofsky, A. F., Kunkel, A. K., Jung, P. G., & Egan, A. M. (2015). Teaching mathematical word problem solving: The quality of evidence for strategy instruction priming the problem structure. *Journal of Learning Disabilities, 48*(1), 51-72. doi: 10.1177/0022219413487408

Jonassen, D. H. (2000). Toward a design theory of problem solving. *Educational technology research and development*, *48*(4), 63-85. Retrieved from https://link.springer.com/article/10.1007/BF02300500

Jonassen, D. H., Howland, J., Moore, J., & Marra, R. M. (2003). *Learning to solve problems with technology*. Pearson Education.

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

Kalelioglu, F., & Gulbahar, 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. Retrieved from https://www.mii.lt/informatics_in_education/pdf/infe232.pdf

Kanbul, S., & Uzunboylu, H. (2017). Importance of coding education and robotic applications for achieving 21st-century skills in North Cyprus. *International Journal of Emerging Technologies in Learning (iJET)*, *12*(01), 130-140. doi: 10.3991/ijet.v12i01.6097

Karaman, G., & Karaman, U. (2019). Comparison of Informatics Technologies and Software Development Course Curricula in 2012 and 2017. *Kastamonu Education Journal*, *27*(1), 309-318. doi: 10.24106/kefdergi.2543

Karna-Lin, E., Pihlainen-Bednarik, K., Sutinen, E., & Virnes, M. (2006). Can robots teach? Preliminary results on educational robotics in special education. In *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)* (pp. 319-321). IEEE. doi: 10.1109/ICALT.2006.1652433

Kátai, Z. (2015). The challenge of promoting algorithmic thinking of both sciences-and humanities-oriented learners. *Journal of Computer Assisted Learning*, *31*(4), 287-299. doi: 10.1111/jcal.12070

Kazakoff, E. R. (2014). *Cats in Space, Pigs that Race: Does self-regulation play a role when kindergartners learn to code?* (Doctoral dissertation, Tufts University). Retrieved from http://ase.tufts.edu/devtech/Theses/EKazakoff_2014.pdf

Kereluik, K., Mishra, P., Fahnoe, C., & Terry, L. (2013). What knowledge is of most worth: Teacher knowledge for 21st century learning. *Journal of Digital Learning in Teacher Education*, *29*(4), 127-140. Retrieved from https://files.eric.ed.gov/fulltext/EJ1010753.pdf

Kotthoff, L. (2016). Algorithm selection for combinatorial search problems: A survey. In *Data Mining and Constraint Programming* (pp. 149-190). Springer, Cham. doi: 10.1007/978-3-319-50137-6_7

Lazakidou, G., & Retalis, S. (2010). Using computer supported collaborative learning strategies for helping students acquire self-regulated problem-solving skills in mathematics. *Computers & Education*, *54*(1), 3-13. doi:10.1016/j.compedu.2009.02.020

Lechelt, Z., Rogers, Y., Yuill, N., Nagl, L., Ragone, G., & Marquardt, N. (2018, April). Inclusive computing in special needs classrooms: designing for all. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (p. 517). ACM. doi: 10.1145/3173574.3174091

Lee, Y. J. (2011). Scratch: Multimedia programming environment for young gifted learners. *Gifted Child Today*, *34*(2), 26-31.

Retrieved from https://journals.sagepub.com/doi/pdf/10.1177/107621751103400208

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. ,& Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads*, *2*(1), 32-37. Retrieved from https://users.soe.ucsc.edu/~linda/pubs/ACMInroads.pdf

Malik, S. I., Mathew, R., & Hammood, M. M. (2019). PROBSOL: A web-based application to develop problem-solving skills in introductory programming. In *Smart Technologies and Innovation for a Sustainable Future*, 295-302. doi: 10.1007/978-3-030-01659-3_34

Mechling, L. C., & Ortega-Hurndon, F. (2007). Computer-based video instruction to teach young adults with moderate intellectual disabilities to perform multiple step, job tasks in a generalized setting. *Education and Training in Mental Retardation and Developmental Disabilities*, *42*(1), 24. Retrieved from http://daddcec.org/portals/0/cec/autism_disabilities/research/publications/_education_training_development_disabilities/2007v42_journals/etdd_200703v42n1p024-037_computer-based_video_instruction_teach_young_adults.pdf

Miller, P. (2009). Learning with a missing sense: What can we learn from the interaction of a deaf child with a turtle?. *American Annals of the Deaf*, *154*(1), 71-82. Retrieved from https://www.jstor.org/stable/26234580

Mohaghegh, D. M., & McCauley, M. (2016). Computational thinking: The skill set of the 21st century. *International Journal of Computer Science and Information Technologies*, *7*(3), 1524-1530. Retrieved from https://unitec.researchbank.ac.nz/bitstream/handle/10652/3422/ijcsit20160703104.pdf?sequence =1&isAllowed=y

Ormrod, J. E. (2000). *Educational psychology: Developing learners. 3rd ed*. Merrill Prentice Hall.

Özsoy, G., & Ataman, A. (2017). The effect of metacognitive strategy training on mathematical problem solving achievement. *International Electronic Journal of Elementary Education*, *1*(2), 67-82.

Polya, G. (1973). *How to solve it.* Princeton NJ: Princeton University Press.

Plerou, A., & Vlamos, P. (2016). Algorithmic thinking and mathematical learning difficulties classification. *Am. J. Appl. Psychol*, *5*(5), 22. doi: 10.11648/j.ajap.20160505.11

Prajapati, R. K., Sharma, B., & Sharma, D. (2017). Significance of Life Skills Education. *Contemporary Issues in Education Research*, *10*(1), 1-6. Retrieved from https://files.eric.ed.gov/fulltext/EJ1126842.pdf

Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science*, *45*(5), 583-602. doi: 10.1007/s11251-017-9421-5

Ratcliff, C. C., & Anderson, S. E. (2011). Reviving the turtle: Exploring the use of logo with students with mild disabilities. *Computers in the Schools, 28*, 241–255. doi:10.1080/073805 69.2011.594987

Reynolds, A. J., Temple, J. A., Ou, S. R., Arteaga, I. A., & White, B. A. (2011). School-based early childhood education and age-28 well-being: Effects by timing, dosage, and subgroups. *Science,* 1203618, 360-365. doi:10.1126/science.1203618

Root, J., Saunders, A., Spooner, F., & Brosh, C. (2017). Teaching personal finance mathematical problem solving to individuals with moderate intellectual disability. *Career Development and Transition for Exceptional Individuals, 40*(1), 5-14. doi: 10.1177/2165143416681288

Salter, J. (2013). Coding for kids: schoolchildren learn computer programming. *The Telegraphy*, *27*, 2014.

Schalock, R. L., Borthwick-Duffy, S. A., Bradley, V. J., Buntinx, W. H. E., Coulter, D. L., Braig, E. M., . . . Yeager, M. H. (2010). *Intellectual disability: Definition, classification, and systems of support* (11th ed.). Washington, DC: American Association on Intellectual and Developmental Disabilities.

Smith, C. C., Cihak, D. F., Kim, B., McMahon, D. D., & Wright, R. (2017). Examining augmented reality to improve navigation skills in postsecondary students with intellectual disability. *Journal of Special Education Technology*, *32*(1), 3-11. doi: 10.1177/0162643416681159

Sternberg, R. (2003). *Cognitive psychology.* Thomson, Wadsworth.

Sternberg, R. J., & Sternberg, K. (2012). *Cognitive psychology*. Wadsworth.

Taylor, M. S. (2018). Computer programming with Pre-K through first-grade students with intellectual disabilities. *The journal of special education*, *52*(2), 78-88. doi: 10.1177/0022466918761120

Taylor, M. S., Vasquez, E., & Donehower, C. (2017). Computer programming with early elementary students with Down syndrome. *Journal of Special Education Technology*, *32*, 149– 159. doi:10.1177/0162643417704439

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

Van Laar, E., van Deursen, A. J., van Dijk, J. A., & de Haan, J. (2017). The relation between 21st-century skills and digital skills: A systematic literature review. *Computers in human behavior*, *72*, 577-588. doi:10.1016/j.chb.2017.03.010

Virnes, M., Sutinen, E., & Kärnä-Lin, E. (2008, June). How children's individual needs challenge the design of educational robotics. In *Proceedings of the 7th international conference on Interaction design and children* (pp. 274-281). ACM.

Voogt, J., Erstad, O., Dede, C., & Mishra, P. (2013). Challenges to learning and schooling in the digital networked world of the 21st century. *Journal of computer assisted learning*, *29*(5), 403-413. doi: 10.1111/jcal.12029

Wainer, J., Ferrari, E., Dautenhahn, K., & Robins, B. (2010). The effectiveness of using a robotics class to foster collaboration among groups of children with autism in an exploratory study. *Personal and Ubiquitous Computing*, *14*(5), 445-455. doi: 10.1007/s00779-009-0266-z

Wang, D., Han, H., Zhan, Z., Xu, J., Liu, Q., & Ren, G. (2015). A problem solving oriented intelligent tutoring system to improve students' acquisition of basic computer skills. *Computers & Education*, *81*, 102-112. Retrieved from https://www.sciencedirect.com/science/article/pii/S0360131514002231

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 215-220). doi: 10.1145/2157136.2157200

Werner, L., Denner, J., & Campe, S. (2014). Using computer game programming to teach computational thinking skills. In *Learning, education and games*, 37-53. Retrieved from https://dl.acm.org/doi/pdf/10.5555/2811147.2811150

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, *366*(1881), 3717-3725. doi:10.1098/rsta.2008.0118

Woods, D. R., Hrymak, A. N., Marshall, R. R., Wood, P. E., Crowe, C. M., Hoffman, T. W., Wright, J. D., Taylor, P. A., Woodhouse, K. A., & Bouchard, C. K. (1997). Developing

problem solving skills: The McMaster problem solving program. *Journal of Engineering Education, 86*(2), 75-91. Retrieved from https://onlinelibrary.wiley.com/doi/epdf/10.1002/j.2168-9830.1997.tb00270.x

Wurdinger, S., & Qureshi, M. (2015). Enhancing college students' life skills through project based learning. *Innovative Higher Education, 40*(3), 279-286. doi: 10.1007/s10755-014-9314-3

Yu, K. C., Fan, S. C., & Lin, K. Y. (2015). Enhancing students'problem-solving skills through context-based learning. *International Journal of Science and Mathematics Education, 13*(6), 1377-1401. Retrieved from https://link.springer.com/content/pdf/10.1007%2Fs10763-014-9567-4.pdf

Appendix A

Dear Students,

Please write your answers to the asked questions into the empty cell in the same row in the table below.

What exactly is our problem?	
How can we solve the problem? Why do we choose this strategy?	
How do the various pieces of information in the problem fit together? How can we use them?	
How much time, effort, etc. should I put into this problem? At which point should I intensify my concentration and my efforts?	
Am I on true track as I proceed to solve the problem? what extent does the initial process of my plan differ from the way I continue?	
Did I solve the problem correctly? If not Why?	

Choreographing Increased Understanding and Positive Attitudes towards Coding By Integrating Dance

Breeann FLESCH[1]

Camila GABALDÓN[1]

Matthew NABITY[1]

Darryl THOMAS[1]

[1]Western Oregon University

DOI: 10.21585/ijcses.v4i3.109

Abstract

Increasing the inclusion of underrepresented individuals in coding is an intractable problem, with a variety of initiatives trying to improve the situation. Many of these initiatives involve STEAM education, which combines the arts with traditional STEM disciplines. Evidence is emerging that this approach is making headway on this complex problem. We present one such initiative, iLumiDance Coding, which attempts to pique the interest and increase confidence of students in coding, by combining it with a fun and physical activity: dance. The connections between dance and coding, while not immediately obvious, are authentic, and we hypothesize that this approach will increase student comfort level with coding. We used student surveys of attitudes toward coding and a variety of statistical approaches to analyze our initiative. Each analysis showed a positive effect on student comfort level with coding. These results are useful for both educators and researchers since they contribute to a deeper understanding of how to increase interest in coding, which we hope will lead to an increase in representation.

Keywords: STEAM education, coding, increasing representation, dance, coding

1. Introduction

Finding ways to spark interest in groups not immediately drawn to coding has been a challenge in an economy that consistently lacks candidates and diversity for positions that require coding skills. Women and minorities are consistently underrepresented in these areas and several initiatives have emerged to attract both of those groups, with mixed results. It is a problem that begins before students even enter college, often starting in middle school. Youth may determine career paths by the age of 13 (Tai et al., 2006; Bernstein et al., 2019; Shet & Tremblay, 2019). Furthermore, the literature addressing middle school students' engagement and identification with STEM and STEM-related careers indicates that developing STEM-specific individual interest (Staus et al., 2020) and preserving feelings of self-efficacy in STEM during middle school is crucial (Aschbacher et al., 2014; Barmby, Kind, & Jones, 2008; Kang et al., 2019; Haussler & Hoffmann, 2002; Nugent et al., 2015). Partovi (2016) of Code.org reports that high school girls and underrepresented minorities who take AP Computer Science are more likely to major in computer science than those who do not (girls 11% vs 1%; underrepresented minorities 14-17% vs 2%). Increasing participation, though, is not simply a matter of offering coursework. In this paper, we present *iLumiDance and Coding*, which attempts to pique the interest and increase confidence of students in coding, by combining it with a fun and physical activity: dance.

Creating engagement by using music and coding have been explored in a variety of projects and has been shown to be a powerful way to teach and learn computer science content, as well as increase coding self-efficacy (Barate et al., 2017; Bell & Bell, 2018; Horn et al., 2020; Ruthmann et al., 2010). Combining dance and coding has been explored to a lesser extent. Shamir et al. (2019) saw positive results from the MathDance, which allowed student to perform a dance while learning computing concepts but did not explicitly draw on the similarities between dancing and coding. While not a combination that immediately comes to mind, explicitly teaching dance and coding together is a powerful way to create authentic cross-disciplinary connections and spark the interest of learners who might otherwise be reluctant in either domain. Upon closer examination, it becomes clear that the ties between dance are coding are numerous and authentic. Coding is a language, just as the construction of visual and kinetic images are a "language" in dance choreography. The word choreography literally means "dance-writing" from the Greek words for dance and writing. In the most basic sense, coding is a set of instructions given to a computer for execution. Dance choreography parallels coding in that it provides instructions for the human body, a "computer" that is self-aware, conscious. Beyond just providing instructions, dance choreography and coding share many concepts that cross both disciplines. Both choreography and coding are, at their core, sequences of primitive

instructions that are executed in a linear order. They both utilize repetition, whether in an "ABA" ternary form or a "for loop." They also have conditional statements that help them respond to dynamic situations or states. For example, the choreography may instruct one dancer to execute a solo movement sequence until within a foot of a second dancer, after the music reaches a certain point both dancers lock arms and revolve around a central point or swing your partner round and round. The moonwalk made popular by the late Michael Jackson, provides an excellent mirror to the use of "if-then" statements in coding. The choreography may instruct the dancer to moonwalk until reaching the edge of the performance space, then execute a 540-degree clockwise spin and continue to moonwalk in the opposite direction. Finally, in both coding and choreography we create "functions" by naming sets of instructions or dance moves (e.g. running man, waltz, polka, twerking, electric slide), which can then be called time and again, without having to recreate the instructions from scratch.

With all of these authentic connections, we conjecture that teaching students coding through dance will increase their comfort level with coding.

2. Literature Review

In recent years, the economy has shown an ever-increasing need in STEM fields, for which the number of qualified applicants is consistently insufficient (Yankman, 2008). In addition to an insufficient number of applicants, employers are also seeking creativity and innovation in new hires that may require them to pull knowledge from more than the discipline in which an employee might be trained (Land, 2013). STEAM education, which expands traditional STEM education to include the arts as a way to foster creative thinking, has emerged as an important pedagogy and may help meet these needs. As employers have noted, complex problems often require using a multi- or trans-disciplinary approach, which is implicit in STEAM education. The process of integrating arts and sciences encourages learners to demonstrate adaptive critical thinking as they hone their ability to develop flexible problem-solving models. The representation of subjects from both the arts and sciences as equals in STEAM education inspires open-ended creative exploration and serves as a form of productive play and inquiry (Trowsdale, 2016).

The interest in STEAM has further increased with the release of the Next Generation Science Standards (NGSS) in 2013, as these standards are designed to teach students to "think critically, analyze information, and solve complex problems" (NGSS, 2013). One NGSS that can be addressed specifically through STEAM projects with coding components is "Using Mathematics and Computational Thinking." Computational thinking is an approach to problem

solving that is based on the fundamentals of computing, and this approach is transforming the way we think about many disciplines, including the natural sciences (Bundy, 2007).

The idea of crosscutting concepts, concepts that are applicable across the domains of science, are also a core part of the NGSS (NGSS, 2019). In many STEAM applications, crosscutting between the arts and sciences is added. Science the language and concepts around science topics can be intimidating to students (Graham and Brouillette, 2016). Likewise, many students are initially intimidated by coding concepts, so, presenting them in a way that shows their connection to the arts has been shown to increase students' comfort level with coding (Moore et al., 2016).

STEAM education not only creates cross-disciplinary connections that can lead to greater creativity, but it also can increase retention of content in within-discipline content. Graham and Brouillette (2016) found that the introduction of arts content into STEM education in elementary schools increased test scores in science areas by 13 percentiles compared to the control group. It has also been shown that by adding the "A" to STEM, we can bring more underrepresented communities into STEM, which continues to be an intractable problem. According to the National Center of Science and Engineering Statistics (NCSES, 2019), of the bachelor's degrees awarded in Computer Science in 2016, only 18.7% were awarded to women and 21.6% were awarded to underrepresented minorities. There are no doubt many causes of this complex problem, but one survey found that women do not think the STEM fields are as interesting as other fields (Weinberger, 2005). A powerful approach to combat this can be through integrating STEM into fields in which girls and women are already interested. For example, STEAM projects that emphasize aesthetics and creativity have been shown to increase participation by young women and girls (Buechley and Mako Hill, 2010). Buchholdz, Shivley, Peppler, and Wohlwend (2014) discovered in their mixed-gender dyads that the "traditional" gender scripts for computing were disrupted by replacing traditional circuitry materials with conductive thread, fabrics, and needles, providing girls the opportunity to take on leadership roles in completing highly complex electronics projects. Another study showed an increase in interest in STEM in a Native American population by using culturally responsive STEAM projects (Kant, Burckhard and Meyers, 2018). Plus, integrating arts into STEM can make the STEM fields more approachable to all, including underrepresented groups (Maltas, 2015).

The use of STEAM projects to learn coding and electronics is not an entirely new concept. For example, the EarSketch project, which integrates music with coding, found that "results reveal that students' attitudes positively and statistically significantly increased across all constructs in our Student Engagement survey, which included constructs such as computing confidence, computing enjoyment, computing perceived usefulness, motivation to succeed in computing, identity and belonging in computing, and intention to persist" (Moore et al., 2016). Several

other programs have also shown strong positive results when connecting music to computer science (Barate et al., 2017; Bell & Bell, 2018; Horn et al., 2020). The Performatics program for college students described by Ruthmann et al. (2010), utilized scratch to teach music students coding concepts and computational thinking while they remixed and created musical pieces. Shamir et al. (2019) showed an increased interest in STEM classes and careers after students combined, music, dance, and animation with computer science. Peppler (2013) took another STEAM approach and utilized e-textiles with students. As part of this, they noted that "Stitching circuits seem to demystify ideas that can be elusive to students using traditional electronics."

3. Methods

While there are multiple STEAM approaches that can be used to introduce coding, our project focuses specifically on the use of dance. It involves the delivery of curriculum utilizing "puppets" and costumes that light up, laptops with our software installed and a connection between them, which currently runs via WiFi over a local router. Students receive some dance and coding instruction but are allowed plenty of room for exploration and creativity in what they design. Our project is founded in the constructivist theories of education, which dates back to Dewey (1934) and even earlier thinkers. In a constructivist framework, students are encouraged to explore, create and play to construct their own understanding of the world. STEAM education is well aligned to fulfill the tenets of constructivist education, where “students are engaged and central to knowledge productions” (Gross and Gross, 2016).

In 2013, Peppler introduced “eight guiding principles of STEAM-powered computing education,” most of which we utilize in our curriculum and delivery. These principles are:

1. Choose open-ended, personal, and aesthetic tools and materials,
2. Make design thinking central,
3. Create authentic combinations of STEM and the arts,
4. Facilitate easy-entry, but challenging, designs,
5. Purposefully contrast multiple media, tools, and materials,
6. Involve a range of disciplinary experts,
7. Devise new assessments, pedagogy, and learning environments,
8. Document and showcase work.

As we described in the introduction, the connections between dance and code are many and authentic. During the course of our STEAM instruction, we explicitly reiterate these authentic connections as we are working on projects, meeting the third principle. Our software is based on Google's Blockly. The colorful, block-based interface provided by Blockly alleviates some

of the "intimidation factor" of coding and allows for easy entry into basic coding concepts, while leaving plenty of room for challenge and creativity, meeting the fourth principle. Students spend most of their time working with the coding software, the puppets and the choreography, with some short, direct instruction. For example, students explore the use of functions and loops in their code on their own, using these tools to develop their desired performance look. This method also follows Peppler's second principle, "make design thinking central." Students are doing this when they try some code, test it to check the visual impact, and then use that as feedback to further adjust their code. They are designing a visual impact in collaboration with designing movement, all the while constructing knowledge about the elements of coding and computational thinking.

Computational thinking, an important 21st century skill, is a core part of the curriculum. And, according to Wing (2018), "the essence of computational thinking is abstraction." In our curriculum, students get experience with abstraction in a variety of ways. For example, students will work on a function, say called lightingChase, where the lights turn on and off in rapid succession. Once they get lightingChase just right, then they can call that function as many times as needed, creating a layer of abstraction. Furthermore, students need to grapple with the abstraction of presentation - between the computer screen and the physical puppets or costumes.

3.1 Research Design

Our curriculum is flexible; we can teach a 5-hour workshop, a week-long camp or even a 10-week course with our software, and it can be adapted for students as young as 4th grade all the way up to young adults. In the 5-hour workshop, students will learn the basics of coding, including the sequential nature of executing code and simple iteration through looping. They will then perform a dance that is choreographed ahead of time but will design the lighting that accompanies the dance using their coding skills. In longer courses, students also learn about conditional statements and functions. They dive deeper into the fundamentals of movement and expression through choreography and develop their own dance(s) with accompanying lights.

When possible, our entire content delivery team is present, which consists of a librarian and professors of dance and computer science, each of whom brings disciplinary expertise and perspective that is unique. This follows Peppler's (2013) sixth principle, "Involve a range of disciplinary experts" and, while one of us will typically be the leader of the workshop, all of us assist and may present pieces as our content areas are discussed.

In the workshop format, the students are split into six groups of six to program six puppets. Within those groups, each student is assigned a laptop to code on and a body part (head, torso,

right arm, right leg, left arm, and left leg), which they will program to light up using Blockly. Each part has six differently colored EL (electro-luminescent) wires that can be programmed as solid, flickering, or off and two LED strips that can produce a custom color based on RGB values the student picks. The laptops and puppets are connected to a local network via a home router system. The EL wires and LED strips are controlled via WiFi using a 12 channel WiFi DC dimmer. For much of the workshop, the students focus on learning how to control the different lights using individual commands, loops, and functions that they design in Blockly. Blockly then produces JavaScript, which both runs the puppets on the laptop screens and sends commands to the individual puppet pieces and wires via the routers. At the end of the workshop, they are asked to coordinate both the movement and lighting of their puppet piece to music in conjunction with their table-mates to create a performance. This follows Peppler's (2013) eighth principle "Document and showcase work."

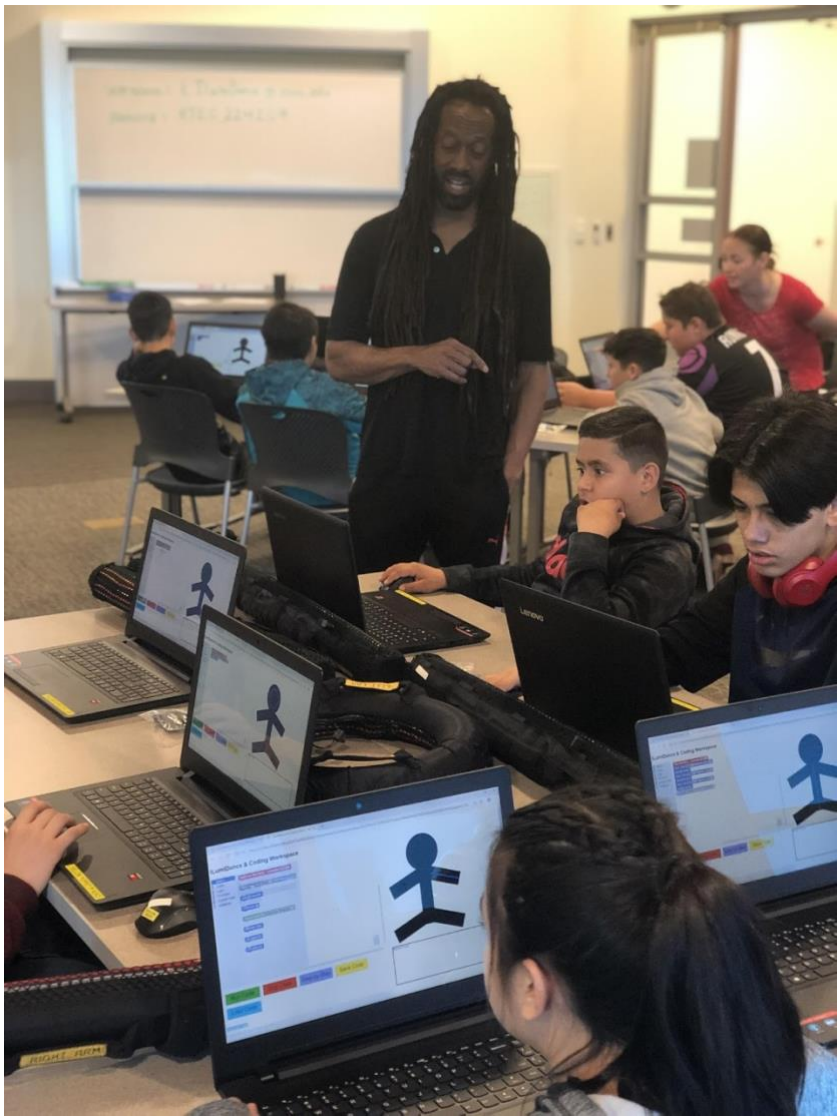


Figure 1. Two team members working with students

In the summer camp, students designed a costume, with separate EL wires in each appendage, which are also controlled via the code they design in Blockly. Like the puppets, the costumes are controlled using a WiFi DC dimmer, but instead of controlling six individually hued wires, each channel controls a single wire representing each body part. The students are asked to bring dark colored clothing, which they use as their costume base. Each is given six pieces of EL wire and six pieces of mesh that they can thread the EL wire through in a design pattern that they choose before attaching the mesh to the appropriate part of their costume. Students then design a lighting sequence and choreograph dancing using Blockly. They may then combine their choreography and lighting for a group dance with shared lighting effects that may appear to create motion or other effects.

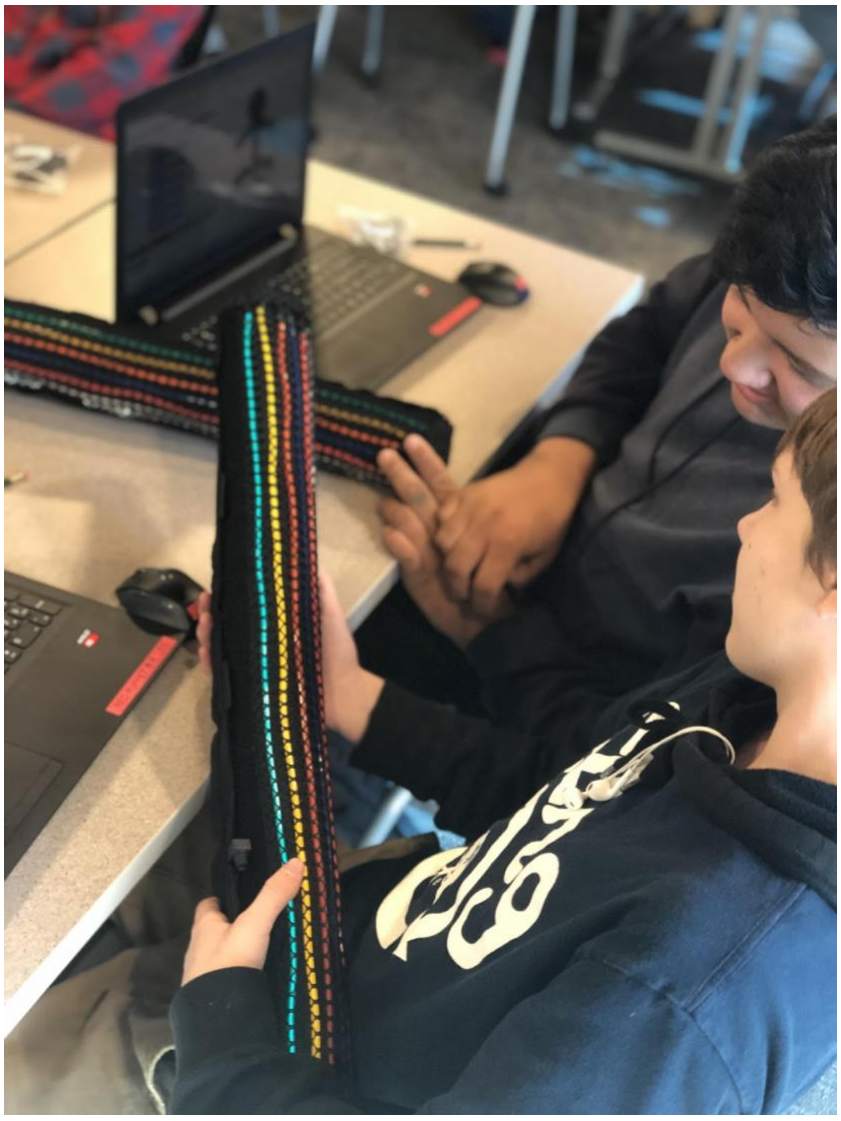


Figure 2. Students programming a puppet piece during a workshop

The 10-week course is still in development but will incorporate additional detailed lessons on coding and movement and the connections between them. Because it is a college-level course, students will also be researching a social issue of their choosing and developing a dance with coded lighting to highlight their representation of that issue. Readings will include articles and chapters on topics such as cultural intersections of dance, art, communication, and technology.

3.2 Data Collection

After each course, we distribute a survey on the workshop to the participants that have given, or whose guardians have given, informed consent, based on our institutional review board approval. Because of limited time, we gave the survey once, at the end of the workshop, and then we chose to maintain a consistent survey method for the summer camp. Other data collection methods, like exit interviews or focus groups, where not used in order to maximize the amount of time for the activity itself, as these students had a strict schedule be get bussed back to their home school. This survey asks workshop participants questions about their attitude about coding before, during and after the survey. For example, they are asked "After participating in the workshop, my comfort level with coding is…." and asked to circle numbers on a Likert scale 1 through 5, with 1 being associated with 'Lower', 3 being associated with 'The Same' and 5 being associated with 'Higher.' Since the 5-hour workshops are for students, there are often other educators in the room (their classroom teachers, paraprofessional educators, counselors, etc.). When this is the case, we also give them a survey that measures their perspective of computing before, during and after the workshop. For example, one question is "I plan to incorporate more coding into my education environments", and they are asked to circle a number on a Likert scale of 1 through 5 with 1 associated 'Not at all,' 3 associated with 'Maybe,' and 5 associated with 'Most Definitely.' These surveys were based on Heidi's (2013) work with measuring effectiveness of Injecting Computational Thinking into Computing Activities for Middle School Girls, which allowed us to use an existing, validated tool.

3.3. Participants

The workshop from which we collected data had students from Prineville, Oregon in Crook County. According to the Oregon Department of Education, 61% of the students from this district are on free and reduced-price lunch, 16% of the students identify as Hispanic and 6% as multiracial (https://www.ode.state.or.us/data/reportcard/reports.aspx). The summer camp pulls students from the Salem-Keizer School District, which has 71% of students on free and reduced-price lunch, 40% identify as Hispanic and 5% as multiracial (https://www.ode.state.or.us/data/reportcard/reports.aspx).

4. Results

The results indicated that combining coding and dance is an effective way of increasing interest, efficacy, and engagement in coding. Participants reported increased confidence with coding, feelings of creativity, playfulness and engagement after the workshops or summer camps.

4.1 Data Analysis

The survey results were used to test our hypothesis that combining dance with coding has positive effect on student comfort level with coding. The questions that we used to measure this comfort level were:

- Question 3: "After participating in the workshop, my comfort level with coding is…"
- Question 4: "If I choose to, in the future I can do a job that uses coding."
- Question 5a: "When I worked on coding the puppets, I felt confused"
- Question 5b: "When I worked on coding the puppets, I felt creative"
- Question 5c: "When I worked on coding the puppets, I felt playful"
- Question 5d: "When I worked on coding the puppets, I felt bored"

Summaries of the data from the spring workshop and the summer camp are reported in Table 1 and Table 2 respectively, and all data is aggregated in Table 3. In addition to reporting the actual data for the survey questions (3, 4, 5a, 5b, 5c, 5d), we compared the survey results in each to randomly completed surveys. As a first assumption for comparison, we considered each of the five responses to be equally likely so that the underlying distribution is uniform. This means the probability of observing any individual outcome from {1,2, 3, 4, 5} is ⅕. Using this, we defined a success as follows:

X= success if we observe a 4 or a 5 on questions 3, 4, 5b, 5c,

or X= success if we observe a 1 or a 2 on questions 5a, 5d,

which establishes underlying binomial distributions for the various values of *n*, the number of survey respondents. From a probabilistic point of view, this means we should expect 40% of the survey results in the respective categories we define as a success. The results in Table 1 suggest that success rate is much higher than predicted for all but Question 4. Looking at the estimated standard deviation in each case, $\sigma_X \approx 2.3$ and $\sigma_X \approx 2.4$ respectively, the observed number of successes is noticeably higher for all but Question 4 as the observed results are nearly two standard deviations or more above what theory predicts. This suggests either the underlying distribution is not uniform or that there was a positive effect on student comfort level with

coding from the workshop. Similar trends are apparent in the summer camp data and the aggregated data.

Likert data in the context of education, specifically STEM disciplines, has received renewed interest (Brickman and Lovelace, 2013). A typical assumption for Likert data is that the data is normally distributed. *"Likert (1932) himself argued that the distances of scores such as 1, 2, 3, 4, 5 are equal and yield data which are approximately normally distributed. The question is if the same is true for the labels (e.g. 'Strongly Agree') which are frequently used. Recent research suggests that literacy affects the ability to discriminate between categories, i.e. that the suitability of the classical Likert scale depends on the choice of the sample (Chachamovich et al., 2009)."* (Treiblmaier and Filzmoser, 2009).

We revisited the calculations in Table 1-3 using various approximate normal distributions. Comparing the expected number of successes with the observed data in Tables 1-3, we found that the observed number of successes was now higher than expected for all questions. The theoretical standard deviations decrease as well making the observed data stand out a bit more than before. Assuming approximate normal distributions indicates the positive effect from the workshop or summer camp was more noticeable. To investigate the possible effect from the workshop and summer camp, we now turn to an inferential analysis of the data to test the hypothesis that the interventions had positive effects.

Table 1. Spring Workshop Summary Survey Results

Question	Predicted # of Successes	Observed # of Successes	n	$\frac{x}{n}(\%)$	$\lvert X_{obs} - X_{pred} \rvert$	Difference from Predicted (%)
3	9.2	17	23	73.9	7.8	33.9
4	9.6	8	24	33.3	1.6	6.7
5a	9.6	14	24	58.3	4.4	18.3
5b	9.6	16	24	66.7	6.4	26.7
5c	9.2	17	23	73.9	7.8	33.9
5d	9.6	17	24	70.8	7.4	30.8

Table 2. Summer Camp Summary Survey Results

Question	Predicted # of Successes	Observed # of Successes	n	$\frac{x}{n}$(%)	$\lvert X_{obs} - X_{pred} \rvert$	Difference from Predicted (%)
3	5.6	8	14	57.1	2.4	17.1
4	5.6	5	14	35.7	0.6	4.3
5a	5.6	6	14	42.9	0.4	2.9
5b	5.6	12	14	85.7	6.4	45.7
5c	5.6	11	14	78.6	5.4	38.6
5d	5.6	9	14	64.2	3.4	24.3

Table 3. Spring Workshop Summary Survey Results

Question	Predicted # of Successes	Observed # of Successes	n	$\frac{x}{n}$(%)	$\lvert X_{obs} - X_{pred} \rvert$	Difference from Predicted (%)
3	14.8	25	37	67.6	10.2	27.6
4	15.2	13	38	34.2	2.2	5.8
5a	15.2	20	38	52.6	4.8	12.6
5b	15.2	28	38	73.7	12.8	33.7
5c	14.8	28	37	75.7	13.2	35.7
5d	15.2	26	38	68.4	10.8	28.4

4.1.1 One-sided Hypothesis Test

We begin with a simple null hypothesis $H_0: p = 0.4$ and test this against the claim that the workshops had a positive effect so that the alternate hypothesis is $H_1: p > 0.4$ for each of the six questions. Based on our initial analysis, we opted to assume the underlying distribution is uniform and to use the test statistic $Z = \frac{\frac{x}{n} - p_o}{\sqrt{\frac{p_0(1-p_0)}{n}}} \geq z_\alpha$, where α is the approximate level of the test. The results for the six questions along with the computed p-values are given in Table 4. For Questions 3, 5b, 5c, and 5d, we note a statistically significant result for nearly any typical

level of significance. Additionally, the p-value for question 5a is significant depending on the desired level of significance.

Table 4. One-sided alternate hypothesis

Question	p_0	x	n	$\hat{p}$	$Z_{\text{test statistic}}$	p-value
3	0.40	25	37	$\frac{25}{37} = 0.676$	3.42	0.00031
4	0.40	13	38	$\frac{13}{38} = 0.342$	-0.730	0.77
5a	0.40	20	38	$\frac{20}{38} = 0.526$	1.59	0.056
5b	0.40	28	38	$\frac{28}{38} = 0.737$	4.24	0.000011
5c	0.40	28	37	$\frac{28}{37} = 0.757$	4.43	0.0000047
5d	0.40	26	38	$\frac{26}{38} = 0.684$	3.58	0.00017

4.1.2 Two-sided Hypothesis Test

The preceding calculations and analysis used hypothesis tests found in almost any statistics text. In practice, these standard confidence intervals and tests of hypotheses often have poor performance even when the sample size is rather large (Agresti and Caffo, 2000). Agresti and Coull (1998) suggested the "add two successes and two failures" interval given by $\tilde{p} \pm z_{\frac{\alpha}{2}}\sqrt{\frac{\tilde{p}(1-\tilde{p})}{\tilde{n}}}$, where $\tilde{n} = n + 4$ and $\tilde{p} = \frac{X+2}{n+4}$. Careful analysis has shown that this interval performs dramatically better than the standard Wald intervals and their associated hypothesis tests explored earlier in this analysis, even when working with small samples (Agresti and Coull, 1998). This confidence interval corresponds to the test statistics $|Z| = \left|\frac{\tilde{p}-p_0}{\sqrt{\frac{p_0(1-p_0)}{\tilde{n}}}}\right| \geq z_{\frac{\alpha}{2}}, or\ |Z| = \left|\frac{\tilde{p}-p_0}{\sqrt{\frac{\tilde{p}(1-\tilde{p})}{\tilde{n}}}}\right| \geq z_{\frac{\alpha}{2}}$, where the difference in the denominator is somewhat up to the practitioner. Using the latter of these two we compute the results reported in Table 5.

Table 5. Two-sided alternate hypothesis

Question	p_0	x	n	$\tilde{p}$	$Z_{\text{test statistic}}$	p-value
3	0.40	25	37	$\frac{25+2}{37+4} = 0.659$	3.49	0.00048
4	0.40	13	38	$\frac{13+2}{38+4} = 0.357$	-0.58	0.56
5a	0.40	20	38	$\frac{20+2}{38+4} = 0.524$	1.61	0.11
5b	0.40	28	38	$\frac{28+2}{38+4} = 0.714$	4.51	0.0000065
5c	0.40	28	37	$\frac{28+2}{37+4} = 0.732$	4.79	0.0000017
5d	0.40	26	38	$\frac{26+2}{38+4} = 0.667$	3.67	0.00024

In any of the preceding calculations, the results are consistent across the inferential structures. Questions 3, 5b, 5c, and 5d are all statistically significant indicating there is evidence for increased confidence with coding, feelings of creativity, playfulness and engagement after the treatment. The Question of student confusion depends on the desired level of significance but at the traditional level of $\alpha = 0.05$ the results are significant. The effect of combining dance and coding is statistically relevant for certain key features central to student comfort level and merit further study.

5. Conclusions

STEAM education is growing rapidly as an important approach to teaching and retaining students in STEM, especially coding (Graham and Brouillette, 2016). Teaching coding with a STEAM approach increases the excitement and creativity around coding and brings underrepresented groups into STEM (Barate et al., 2017; Bell & Bell, 2018; Horn et al., 2020; Ruthmann et al., 2010, Shamir et al. 2019). However, we feel it is important that the STEAM connections to coding be authentic and transdisciplinary, which is why coding and dance are such a natural fit.

The numerous connections and parallel concepts between coding and dance make it a natural and authentic pairing for instruction. It is approachable to people with a variety of backgrounds or comfort levels with either activity and promotes creativity and playful exploration of both. In each of the venues where we collected data, we measured students' attitudes towards coding as a possible measure of future success as described by Else-Quest, Mineo, and Higgins (2013) and in each case, we found that there was increased comfort with coding, feelings of creativity, playfulness and engagement using this approach. Like Wanzer et al. (2020), we saw an increased feeling of affiliation toward coding and coding activities in students self-reporting after participation in the activities. In addition to the statistical data we gathered, we received lots of enthusiastic verbal feedback from participants. Of particular note, one 8th grade girl commented that "I really was expecting us to do some normal coding, like we do at school. But, this was totally different. It's more hands-on. And I think if we did more of this, more girls would take coding" after the workshop.

While this outcome can't help but be exciting, we must acknowledge some of the limitations in our study. First, our sample sizes are small, and we did not have a control group. Our project is in its early stages, so the data must be viewed as preliminary. Second, our samples, while diverse, were not entirely randomly selected; the summer camp participants self-selected and the workshop participants had teachers who had selected to involve their classes with these activities, which could indicate previous exposure to coding and coding concepts, though does not account for changes in attitude.

Our future work includes using this approach in a quarter-long class at the college level and attempt to measure any change in the understanding of coding basics in addition to attitudes. We will also continue to provide workshops and summer camps and collect data on student attitudes. We hope to start collecting data on race and gender, and as our sample gets bigger, disaggregate the data to further explore underrepresented groups.

Over the course of this project we have met with a variety of individuals from classroom teachers to large tech company executives, and many expressed frustrations about the difficulty of sparking excitement for coding. Based on our results, our project provides one avenue to address that frustration by combining learning coding with a topic that some already feel passionate about. This approach to teaching coding can be implemented by K-8 teachers as well as informal science educators. While our workshops utilized puppets and dedicated laptops, the approach does not require these. Instructors can utilize existing software and platforms, such as Scratch or Blockly and still emphasize the authentic connections between coding and dance or

another activity that combines similar concepts. An example of this might be creating artwork, finger weavings, or music based on coding concepts. Explicitly describing and utilizing computer science concepts, such as iteration, looping, and variables, while doing that activity, then showing how that concept is also used in the context of creating code can make powerful connections in students' minds. The efficacy of pairing coding and dance compared to other combinations of artistic endeavors and coding is unknown, but there may be an advantage to having a variety of STEAM approaches to getting individuals excited about coding. We are enthusiastic about the inroads into coding that these STEAM projects provide and are interested in how the results of other STEAM pairings compare to ours.

References

Agresti, A. and Coull, B.A. (1998). Approximate is better than 'exact' for interval estimation of binomial proportions. *The American Statistician*, 52, p. 119126.

Agresti, A. and Caffo, B. (2000). Simple and effective confidence intervals for proportions and differences of proportions result from adding two successes and two failures, *The American Statistician*. 54, 280288.

Aschbacher, P. R., Ing, M., & Tsai, S. M. (2014). Is science me? Exploring middle school students' STEM career aspirations. *Journal of Science Education and Technology*, 23(6), 735–743. https://doi.org/10.1007/s10956-014-9504-x

Baratè, Adriano, Luca A. Ludovico, and Dario Malchiodi. "Fostering computational thinking in primary school through a LEGO®-based music notation." *Procedia computer science*, 112 (2017): 1334-1344.

Barmby, P., Kind, P. M., & Jones, K. (2008). Examining changing attitudes in secondary school science. *International Journal of Science Education*, 30, 1075–1093.

Bell, J., & Bell, T. (2018). Integrating computational thinking with a music education context. *Informatics in Education*, 17(2), 151-166.

Brickman, P. and Lovelace, M. (2013). Best practices for measuring students' attitudes toward learning science. *CBE life sciences education*. 4. 606617.

Buchholz, B., Shively, K., Peppler, K., and Wohlwend, K. (2014). Hands On, Hands Off: Gendered Access in Crafting and Electronics Practices. *Mind, Culture, and Activity.* (21)4, 278-297. https://doi.org/10.1080/10749039.2014.939762

Buechley, L. and Mako Hill, B. (2010). LilyPad in the wild: how hardware's long tail is supporting new engineering and design communities. *In Proceedings of the 8th ACM Conference on Designing Interactive Systems (DIS '10).* ACM, New York, NY, USA, 199-207. 10.1145/1858171.1858206

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing*. 1(2), 67-69

Dewey, J. (1934). *Art as experience.* Westport, CT: Praeger.

Else-Quest, N., Mineo, C., and Higgins, A. (2013). Math and Science Attitudes and Achievement at the Intersection of Gender and Ethnicity. *Psychology of Women Quarterly.* (3)37, 293-309. https://doi.org/10.1177/0361684313480694

Gross, K. and Gross, D. (2016). TRANSFORMATION: Constructivism, Design Thinking, and Elementary STEAM, *Art Education,* (6)69, 36-43. https://doi.org/10.1080/00043125.2016.1224869

Haussler, P., & Hoffmann, L. (2002). An intervention study to enhance girls' interest, self-concept, and achievement in physics classes. *Journal of Research in Science Teaching*, 39(9), 870–888.

Horn, M., Banerjee, A., West, M., Pinkard, N., Pratt, A., Freeman, J., ... & McKlin, T. (2020). TunePad: Engaging learners at the intersection of music and code.

Kang, H, et al. (2019). How do middle school girls of color develop STEM identities? Middle school girls' participation in science activities and identification with STEM careers. *Science Education,* 103:418–439.

Kant, J., Burckhard, S. and Meyers, R. (2018). Engaging High School Girls in Native American Culturally Responsive STEAM Activities. *Journal of STEM Education*, 18(5). Retrieved September 13, 2019 from https://www.learntechlib.org/p/182466/

Maltas, W. (2015). From STEM to STEAM: Integrating Arts Education into the STEM disciplines of Science, Technology, Engineering and Math. Drexel University, Philadelphia, PA.

Moore, R., Edwards, D., Freeman, J., Magerko, B., McKlin, T., and Xambo, A. (2016). EarSketch: An Authentic, STEAM-based Approach to Computing Education. 2016 American Society for Engineering Education Annual Conference & Exposition.

NGSS Lead States, (2013). *Next generation science standards: For states, by states.*

Nugent, G., Barker, B., Welch, G., Grandgenett, N., Wu, C., & Nelson, C. (2015). A model of factors contributing to STEM learning and career orientation. *International Journal of Science Education*, 37(7), 1067–1088.

Oregon Department of Education. 2019. At-A-Glance School and District Profiles. https://www.ode.state.or.us/data/reportcard/reports.aspx Accessed 28 September 2019.

National Center of Science and Engineering Statistics. https://ncses.nsf.gov/pubs/nsf19304/digest/field-of-degree-women#computer-science. Accessed 28 September 2019

Peppler, K. (2013) STEAM-Powered Computing Education: Using E-Textiles to Integrate the Arts and STEM, *Computer*, 46(9), 38 - 43. DOI: 10.1109/MC.2013.257.

Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P., & Saulters, C. (2010, March). Teaching computational thinking through musical live coding in scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 351-355).

Shamir, M., Kocherovsky, M., & Chung, C. (2019, March). A paradigm for teaching math and computer science concepts in k-12 learning environment by integrating coding, animation, dance, music and art. In *2019 IEEE Integrated STEM Education Conference (ISEC)* (pp. 62-68). IEEE.

Treiblmaier, H. and Filzmoser, P. (2009). Benefits from using continuous rating scales in online survey research, tech. rep., Vienna University of Technology, Augasse 2-6, A-1090 Vienna, Austria, November2009. https://pdfs.semanticscholar.org/6f90/c913e1e5a162371862a6c5e20f8e7de5e9c7.pdf

Wanzer, D. L., McKlin, T., Freeman, J., Magerko, B., & Lee, T. (2020). Promoting intentions to persist in computing: an examination of six years of the EarSketch program. *Computer Science Education*, 1-26.

Webb, H. (2013). *Injecting Computational Thinking into Computing Activities for Middle School Girls, A Dissertation in Information Sciences and Technology from The Pennsylvania State University.* The Graduate School College of Information Sciences and Technology.

Weinberger, C.J. (2004). Just ask! Why surveyed women did not pursue IT courses or careers, *IEEE Technology and Society Magazine* 23, 28-35.

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

Yakman, G., (2008). STEAM Education: An Overview of Creating a Model of Integrative Education, Virgina Polytechnic and State University: Virgina. https://www.iteea.org/File.aspx?id=86752&v=75ab076a

Ineffectiveness of Online Interactive Video Content Developed for Programming Education[2]

Mahmut Can SÖZERİ[1]

Serhat Bahadır KERT[2]

[1]Kuveyt Turk Participation Bank IT R&D Center

[2]Yıldız Technical University

DOI:10.21585/ijcses.v4i3.99

Abstract

In this study, the effects of interactive video usage in programming education on academic achievement and self-efficacy perception of programming were investigated by taking into account learning styles. The research was patterned according to the causal-comparative model, and also, correlation analysis was performed for related research. Sixty-one students attending 3rd grade in Computer and Instructional Technology Education (CEIT) of Yıldız Technical University participated in the study. Research data were collected with "Interview form," "Academic Achievement Test," "self-efficacy scale for programming," "learning styles index" and "interactive Video system." ANCOVA, Correlation and Kruskal Wallis H-test were used in the analysis of the data. The data was analyzed on the computer with the SPSS package program. According to research findings, interactive video monitoring rates did not differ significantly in the students ' academic achievement and self-efficacy perceptions of programming. It was found that students ' learning style preferences had no impact on interactive video viewing rates. The relationship between the change in students ' academic achievement and the change in self-efficacy perception scores related to programming has been examined; as academic achievement increases; it has been concluded that the perception of self-efficacy about programming has fallen.

Keywords: Interactive video; computer science education; learning style; programming; self-efficacy of programming.

[2] This article was produced from master thesis of Mahmut Can SÖZERİ

1. Introduction

With the technology that develops day by day, internet tools have been affected by these developments and have brought up a new concept with them.: Web 2.0 (Akar, 2010). Different researchers offer various definitions of Web 2.0. For example, Tu, Blocher, and Ntoruru (2008) define it as "a Web technology that aims to enhance creativity, information sharing and collaboration among users" (p. 336). Scrum and Levin (2009) emphasize the same features and explain it as the second generation of the internet that supports creativity, information sharing and collaboration. Akar (2010) states that Web 2.0 is a technology that makes the internet more participatory, creative and social. Bustamante (2017) prefers to use the term as one of the key mediums of technology-based teaching and defines it as effective interactive media used to improve students' achievement in the classroom. . Web 2.0 is emerging development to increase the usability and functionality of web technologies (Karaman, Yıldırım & Kaban, 2008). In this research, with the perspective of interaction, Web 2.0 is operationally defined as the technology improving interactivity among all stakeholders of the internet. Some of the Web 2.0 tools can be listed as follows: Social networking systems-Twitter, Facebook, video sharing sites-YouTube, Google video, image sharing sites-Flickr, Instagram, wikis, blogs, virtual worlds, podcasts, interactive video systems (Munoz and Towner, 2009; Brame, 2016). Videos are one of the Web 2.0 technologies, which is becoming very popular and also used frequently (Kolowich, 2016). Additionally, it can be said that there has been an increase in video usage rates in educational contexts (Yıldırım & Özmen, 2012). Improving learner motivation is one of the important reasons for using video content in educational environments. Videos increase the motivation of tech students if used as learning and teaching tools (Duffy, 2008). However, if the non-interactive video was used during the teaching process, an insufficient level of satisfaction was revealed (Kozma, 1986). The reason for this is that video surveillance provides a passive experience as in reading (Brame, 2016). Interactive video (Schaffer & Hannafin, 1986), also known as hyper video (Petan, Petan &Vasiu, 2014), is an instructional technology that combines the computer's ability and capabilities of video, allowing the student to interact with an instructional video from a passive audience position (Zhang, Zhou, Briggs & Nunamaker Jr., 2006). Some of the ways to make a video interactive video are:

- Embedding open-ended and multiple-choice questions on video and giving feedback based on responses (Schaffer & Hannafin, 1986; Yin, Lin, Yang and Chen, 2013),
- Repeating of the specific section on the video according to the answers given to the questions and continuing to this repetition until the correct answer is answered (Schaffer & Hannafin, 1986),
- Displaying chapter titles on video, including sections of topics within the video (Petan, Petan & Vasiu, 2014),

- Showing and drawing text, tables, and images on the video. (Yin, Lin, Yang and Chen, 2013)

Interactive video usage increases attention, remembering, satisfaction and video viewing times (Clothier, 2013). In this context, in academic studies (Barthel, Ainsworth & Sharsples, 2013; Hrastinski & Monstad, 2014; Yin, Lin, Yang & Chen, 2013) it has been observed that video-based systems including technological futures to support user-computer interactions.

In this study, interactive videos were used in the process of graphics-based programming language education. The interactive videos are based on “how-to” videos which are used to teach new skills or new techniques/technologies. It is seen that “how-to” videos are becoming more and more popular in video distribution environments. In the Twenty-First Century, individuals are expected to make production on their own as well as consumption (Kalelioğlu, 2015). Interactive videos can help them to create new things on their own. The value given to the training of individuals who can produce on their own by learning programming is evident (Ersoy, Madran & Gülbahar, 2011; Kukul and Gökçearslan, 2014). The value given to programming in Turkey is increasing day by day (Kert and Uğraş, 2009). Nowadays, various approaches are being used by practitioners to improve the programming skills of the students. This research has been focused on the effect of interactive videos on the success of the students in programming courses. Additionally, Self-efficacy and learning styles of the individuals have been thought of as two important factors affecting video-supported education and therefore added to the investigation of the correlations.

Self-efficacy for programming is one of these areas (Altun and Mazman, 2012). Programming skills are seen as more challenging to understand than university-level courses such as mathematics and science (Aşkar and Davenport, 2009). This is because individuals see programming as a more difficult issue to understand, in other words, their perceptions of self-efficacy about programming are low (Altun and Mazman, 2012). In the field of literature, several studies have been examined between academic achievement and self-efficacy, but it has been observed that there is an inadequate number of studies taking into account the impact of self-efficacy on programming skills (Aşkar and Davenport, 2009). Because of the different mental processes of the students in the research, it has been found that they have configured and identified information differently (Samancı & Keskin, 2007). The self-efficacy of the individual can be seen as an essential factor in the development of programming skills. Self-efficacy is defined as "self-judgment about the capacity of the individual to organize and successfully perform activities necessary to show a certain performance" (Gözüm and Aksayan, 1999). This concept is used by adapting to different disciplines and fields (Seferoğlu & Akbıyık, 2005).

Learning styles affect the structure of learning-teaching environments (Arslan and Aksu, 2006). Therefore, in educational settings, teachers need to have general knowledge about the profiles of learners (Arslan and Aksu, 2006; Felder and Henriques, 1995). There are many inventories developed to identify learning styles. One of them is the learning style index which was developed by Felder and Silverman in 1994 (Felder and Soloman, 1994). This index determines the learning style preference in 4 factors with 44 questions. Felder's learning factors are doing-thinking, feeling-intuitive, visual-verbal, sequential-holistic (Howard, Carver, and Lane, 1996). Before preparing the index of learning style factors and learning styles, Felder examined four learning model styles: The Myers-Briggs Type Indicator(MBTI), Kolb's Learning Style Model, The Herrman Brain Dominance Instrument (HBDI) (Thomas, etc., 2002).

The main aim of the research is to examine the effect of interactive course video support on the process during the programming education process. Given the possible effect of self-regulation skills and learning style of the students it is aimed to find the answers to the following sub-problems:

RQ1: Is there a significant correlation between interactive video monitoring rates and self-efficacy perception scores for programming?

RQ2: Is there any effect of learning styles on video monitoring rates of the students?

RQ3: Is there a significant correlation between students' academic achievement and interactive video viewing rates?

RQ:4 Is there a significant correlation between the change in students' academic achievement and the change in self-efficacy perception scores related to programming?

2. The study

List of Abbreviations

AAT : Academic Achievement Test

CPSES : Self-efficacy Scale For Programming

ILS : Index of Learning Styles

IVMR : Interactive video monitoring rates

IVS : Interactive video system

G : Study Group

ILS : Index of Learning Styles

2.5.Method

The study was designed according to the causal-comparative model, and the correlation research model was used in the research. A causal-comparative research pattern is a pattern used to determine the causes or consequences of differences between groups within existing groups (Fraenkel, Wallen& Hyun, 2011). The correlation model is the model used to determine the relationship between variables (Sönmez & Alacapınar, 2013). In the correlation model, a causal comparison model is needed because no interpretation can be made in the context of the cause-effect model. The causal-comparative research model is the study of "determining the causes that influence the results of a completed case." (Sönmez & Alacapınar, 2013). In this model, the researcher does not insert any variables in the environment, nor does he interfere with the results. Therefore, this model is used to determine the differences between individuals. (Büyüköztürk, Çakmak, Akgün, Karadeniz & Demirel, 2010).

The dependent variables of the research are the perception of self-efficacy in programming and academic achievement. The independent variables are the learning styles, the IVS (interactive video system) monitoring rates, and the number of questions they answer correctly in IVS. Scale and tests were applied to all students in the study group. At the same time, all students had the opportunity to use IVS. The symbolic representation of the study with these explanations is presented in Table 1:

Table 1. The Symbolic Representation Of The Research Model

Group	**Pre-Test**	**Process**	**Post-Test**
G	AAT1 CPSES1 ILS	Conventional Course Use of IVS as a supportive source	AAT2 CPSES2

2.6.Participants

This research was carried out with 66 students in 2 groups who took Multimedia Design and production courses while studying in the computer and instructional technologies Education Department of Yıldız Technical University Faculty of Education. However, during the research process, six students who did not attend the courses regularly and who did not support the studies were excluded, and the study was completed with 60 students. All students who participated in the study were provided with access information for IVS, which was developed by the researcher, and they were allowed to use IVS.

In this study, groupings were made by causal-comparative research pattern: IVS never used/viewers (never viewed), IVS infrequent users/viewers (1.5 hours or fewer), IVS active users/viewers (1.6 hours and more). The reason for the preference of this course is that students

have taken courses in the first grade, second grade, and the first semester of third grade, and have gained fundamental knowledge and skills on this subject. Besides, students who failed to take the course in previous years and who took the course from the upper grade even though they were in a lower grade are not included in the study. Nobody was forced to watch videos throughout the semester. Just, gift cards were used as motivators of the process. In the first week, It has been announced that 10 students with the highest video viewing rates will be rewarded with a gift card from a book store.

Students have taken three hours of Multimedia Design and Production courses per week. The course is given in the classroom environment and by CEIT faculty members. The videos on IVS were obtained weekly by using the Camtasia program at the end of the course, which was conducted by the researcher, and uploaded to IVS. The researcher and the lecturer of the course have provided the recording of the videos in a coordinated way, before and after each course, in the same way as the content of the course. After the videos were taken, the interaction was added and made available to students at IVS.

Table 2. Demographics of the Participants

Age (Year)	**f**	**%**	**Gender**	**f**	**%**
18-20	13	21.67	**Male**	29	48.33
21-25	44	73.3	**Female**	31	51.67
26-35	3	5	**Total**	60	100
Total	60	100			

The study included 60 students, 29 of them are males (48.33%), and 31 of them are females (51.67%). When students' ages were examined, the age distribution was found to be between 18-20 (21.67%) and 21-25 (73.3%). Also, 5% of the students were found to be in the age group of 26 and above. Additionally, In Table 3, The results of the students according to the Felder and Soloman learning styles index are given. In the results, each student is presented in detail as weak-medium-strong according to four factors and two poles of each factor.

Table 3. Learning Styles of Students in 4 Factors

Learning Style	**Efficacy**	**f**	**%**
Doing-Thinking	Doing – Weak	31	50.82
	Doing – Medium	10	16.39
	Doing – Strong	4	6.56
	Thinking – Weak	13	21.31

	Thinking – Medium	3	4.92
	Thinking – Strong	0	0
	Total	**61**	**100**
Intuitive - Feeling	Intuitive – Weak	6	9.83
	Intuitive – Medium	2	3.28
	Intuitive – Strong	0	0
	Feeling – Weak	32	52.46
	Feeling – Medium	19	31.15
	Feeling - Strong	2	3.28
	Total	**61**	**100**
Visual - Auditory	Visual – Weak	21	34.42
	Vısual – Medıum	19	31.15
	Visual – Weak	15	24.60
	Auditory – Weak	3	4.92
	Auditory – Medium	2	3.28
	Auditory - Strong	1	1.64
	Total	**61**	**100**
Sequential - Holistic	Holistic – Weak	15	24.60
	Holistic – Medium	3	4.91
	Holistic – Strong	0	0
	Sequential – Weak	28	45.90
	Sequential – Medium	14	22.95
	Sequential - Strong	1	1.64
	Total	**61**	**100**

As shown in Table 3, students' apparent distributions of learning styles are as follows: 50.82% Doing-Weak, 52.46% Feeling-Weak, 45.90% Sequential-Weak.

2.7.The Data Collection Tools

The data obtained in this study were collected through tests and scales, a. Pre-test, self-efficacy perception scale and learning styles index, which are achievement tests, were applied before the use of IVS. In the process of using IVS, information about using IVS has been collected. After the use of IVS, the self-efficacy scale for programming and final-test was applied.

The “Academic Achievement Test” was developed by the researcher to scale the success of the interactive video system for students in software education. To have a measurement tool validity, “all observed and measurable properties of the desired quality should be present in a

measurement tool." (Sönmez and Alacapınar, 2013). The validity of the academic achievement test has been decided by taking expert opinion from the faculty member of the course. The Achievement Test has been prepared as 40 questions to measure target behaviors. The test was applied to 51 students in the study group. It is not possible to reach consistent and accurate judgment with the data obtained through reliable and non-valid measurement tools (Sönmez and Alacapınar, 2013). In this respect, the reliability of the test was determined by Kuder-Richardson-20 (KR-20) technique. As a result of the application, the KR-20 Reliability Coefficient was calculated as 0.59. For each subject in the Achievement Test, difficulty and differentiation indices were calculated. There are 20 questions in the final of the Achievement Test after the removed substances. KR-20 Reliability Coefficient was calculated as 0.77. The average degree of difficulty of the substances was found as 0.48. The degree of difficulty of the substances in the test varies between 0.21 and 0.79. The test includes four simple questions, nine moderate questions, and seven difficult questions.

To determine learning styles, we used the "Learning Style Index" developed by Felder and Silverman (1988) which is based on the learning style model and scaled by Felder and Soloman (1994) and adapted to the Turkish language by Samancı and Keskin (2007). The web-based version of the Felder-Soloman Learning Style Index has been used by more than 100,000 people each year to determine the learning style and has been involved in many studies (Litzinger, Lee, Wise & Felder, 2005). The validity and reliability of the index were made by Samancı and Keskin (2007) and by Litzinger and colleagues (2005). Reliability Coefficients are shown in Table 4.

Table 4. Reliability Coefficients of Four Learning Styles in The Index

Doing Thinking	**Feeling Intuitive**	**Visual Auditory**	**Sequential Holistic**	**N**	**Study**
0.43	0.54	0.59	0.32	381	Samancı and Keskin (2007)
0.60	0.77	0.74	0.56	572	Litzinger and colleagues (2005)

For all of the learning factors, Samancı and Keskin (2007) found that the correlation between factors was quite close to zero and statistically equal to zero. Litzinger and his colleagues (2005) also stated that each factor corresponds to all the objectives of the scale. Forty-four substances contained in this index are made up of four factors. These factors are Doing-Thinking, Feeling-Intuitive, Auditory-Visual, and Sequential-Holistic. Each factor is matched with 11

expressions. Each expression consists of two options: A and B. According to the number of A and B responses given for each factor, it defines the level of the respective factor to be strong, medium and weak (Samancı and Keskin, 2007).

In this study, the "Self-Efficacy Perception Scale For Programming" developed by Ramalingam and Wiedenbeck (1998) and adapted to Turkish by Altun and Mazman (2012) was used to determine the perception of self-efficacy in programming. The scale developed by Ramalingam and Wiedenbeck (1998) for C++ programming language is used in Java programming language, Aşkar and Davenport (2009). The scale, which was developed in English and consisted of 32 items, was determined by the research conducted by Altun and Mazman (2012), which consisted of 9 items and 2 factors in Turkish form.

2.8.Interactive Video System (IVS) features

IVS is the general name of the system that the learner uses continuously during a teaching process. The student on the IVS performs all transactions and all processes. The administrator/instructor/teacher communicates with this system through the management panel. The overall structure of the system is shown in Figure 1.

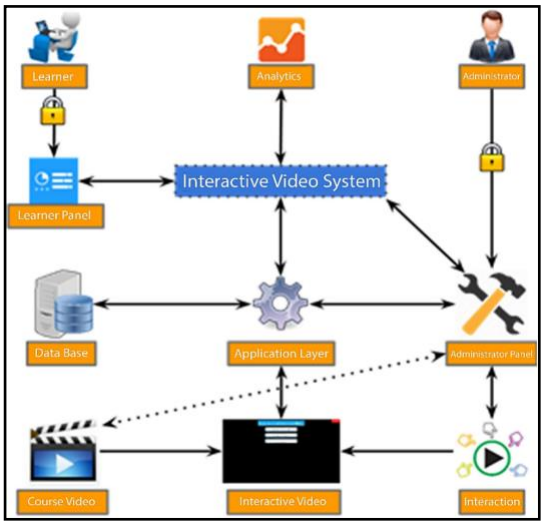


Figure 1. Interactive Video System (IVS) usage scheme

Students entered the IVS system through the student login screen. It is also stated that the student number and password must be in a specific pattern and the warning should be given in case of an incorrect entry. The first screen of the student after logging in to IVS is the IVS Home Screen. This screen contains topics and sub-topics. The middle section shows the video about the sub-topic clicked. The video opens in the middle of the page to fit the screen

resolution, clicking on one of the topics in the left menu. A view of the IVS video display page is shown in Figure 2.

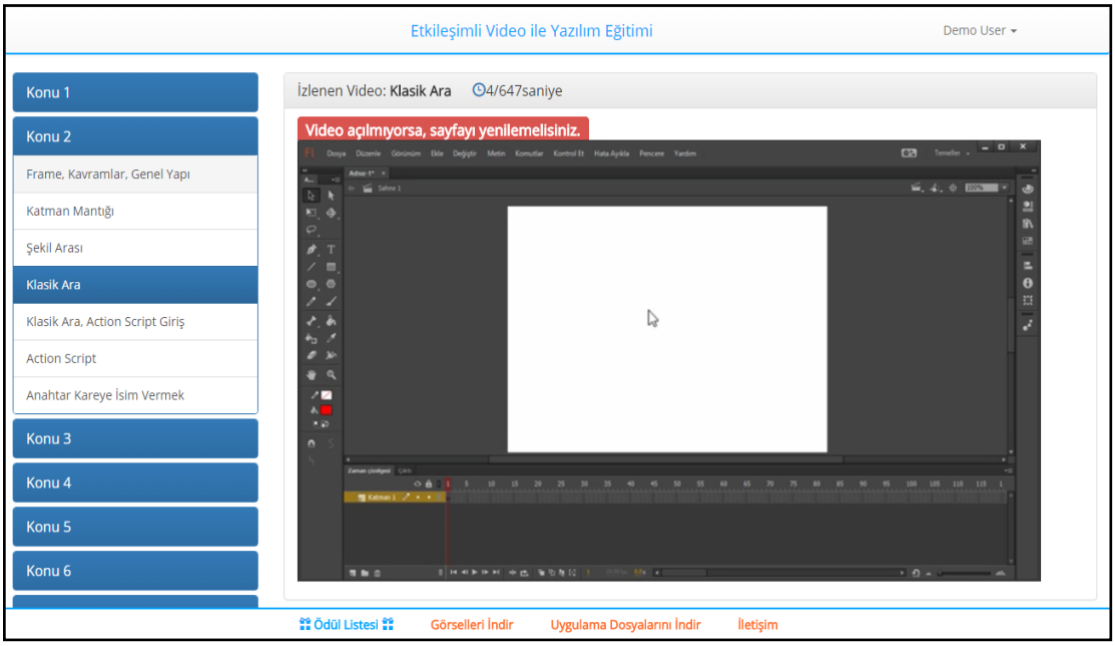
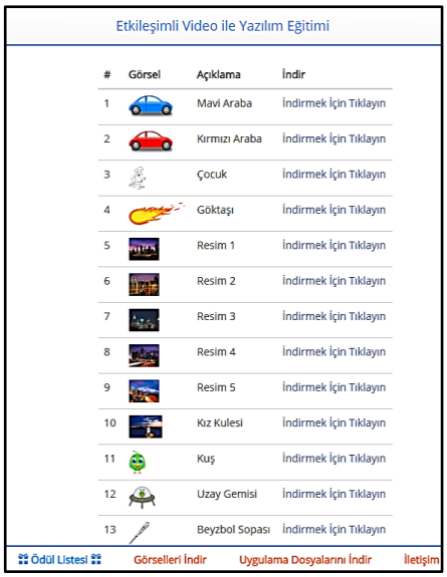
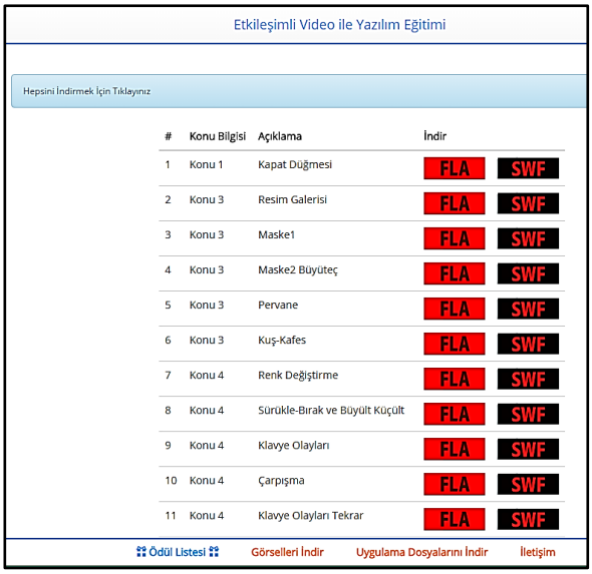


Figure 2. IVS Video Display Page

The student encounters the interaction added by the teacher while watching the video. Here, the student can respond or continue to follow without responding. The images used in the samples in the videos are downloaded through the screen, shown in the screenshot below, which is opened by clicking on the “download images” link. The application files of the samples developed in the videos can be downloaded from the “download application files” screen shown in the screenshot below. The views for the file download pages are presented in Figure 3.





Downloading images used in samples Downloading sample files

Figure 3. File download pages on IVS

The management panel of IVS is the administration section of the system. A view of the IVS management panel is presented in Figure 4.

Figure 4. IVS Management Homepage

In this screen, the number of students currently using the application, answers to questions, total video time, number of entries, number of videos watched per student, number of students watching video per day, number of entries per day and number of entries per student are

statistically shown. In addition to this, the administrator can do the following: changing students log in information; viewing, listing, adding, deleting, and updating courses; viewing, listing, adding, deleting, and updating videos; viewing, listing, adding, deleting, and updating interactions; viewing, listing, adding, and updating students; adding, deleting, and updating students.

2.9.Data Analysis

The SPSS package program was used to analyze research data. When interpreting results, $p \leq .05$ as a level of significance accepted. Information about five students collected through interview form is presented in the study by correcting spelling errors. No coding or additional commentaries have been made. First of all, parametric and non-parametric controls were performed in the analysis of the data. A Single Factor Covariance Analysis (One Factor ANCOVA) was used to analyze whether the change in self-efficacy perception scores for programming differs significantly compared to interactive video monitoring rates. The Kruskal Wallis(KW) H-test was used to analyze whether interactive video monitoring rates differ according to students' learning style preferences. Correlation is used to interpret the amount and direction of the association between variables (Büyüköztürk, 2014). A correlation was used to examine the association between the amount of change in students' academic achievement and the amount of change in perceptions of self-efficacy regarding programming.

"In planning a study, researchers have the responsibility to evaluate carefully any ethical concerns." (Fraenkel, Wallen & Hyun, 2011, p. 62). The whole process was carried out in compliance with the ethical standards of scientific research. First of all, it can be mentioned that the underlying reason for using causal-comparative design instead of an experimental one in the research was avoiding possible ethical issues. The authors wanted to deliver interactive videos to all students instead of only those in the experiential group. Any part of the process was not manipulated by the researchers. Watching the videos was not compulsory and entirely up to the students' decision. Additionally, the students did not see each other's follow-up times. The names of the students were not used in any part of the research. The consent of the participants was obtained for the user data of the research.

3. Results

In this section, the data obtained using the data collection tools, together with the results of the analysis of this data through the appropriate statistical methods and comments on these findings are included. When interpreting the findings, $p<.05$ was considered as a level of significance.

3.1 Self-efficacy perception

Interactive video monitoring rates were obtained in hours/seconds by IVS. Twenty students who did not fill one or both of the self-efficacy scales for programming (CPSES) fill missing or write their name on the scales were excluded. The analysis was carried out with 40 student data (N=40) as shown in Table 5.

Table 5. Grouping of Interactive Video Monitoring Rates

Group No	**Including Criteria**	**Condition**	**N**
Group 0	Who Never Viewed The Interactive Video	Never viewed	12
Group 1	Who Viewed The Interactive Video Infrequently	Who watches for 1.5 hours or less	15
Group 2	Who Viewed The Interactive Video Actively	Who watches 1.6 hours or more	13

Since the group capacity is less than 50, the Shapiro-Wilks Test (Büyüköztürk, 2014) was used for normality analysis. According to the results, p-values were greater than a=.05 can be interpreted as the scores did not show a significant deviation from a normal distribution (Büyüköztürk, 2014). The normal distribution of data in the self-efficacy perception scale for programming has shown that parametric analysis methods can be used in data analysis (Büyüköztürk, 2014). A Single Factor Covariance Analysis (One Factor ANCOVA) was used to analyze whether there was a significant correlation between interactive video monitoring rates (independent variables) and CPSES final application scores (dependent variables). CPSES pre-application scores were used as a covariate. Explanations of the Covariance Analysis Results are shown in the table below.

Table 6. CPSES Final Application Scores Covariance Analysis Results

Source	Sum of Squares	Degree of Freedom	Mean of Squares	F	p
CPSES Pre-Application	2649.848	1	2649.848	75.838	.000
Group	178.209	2	89.105	2.550	.092
Bug	1257.879	36	34.941		

When table 6 was examined; It was inferred that CPSES scores did not differ significantly from the interactive video monitoring rates (groups), ($F(2, 36) = 2.550$; $p > .05$). In other words, the perception of self-efficacy of programming has no relation to interactive video monitoring rates. Correlation between the amount of change in students' academic achievement and their perception of self-efficacy related to programming were analyzed using a simple correlation technique. Results are shown in Table 7.

Table 7. Results of Correlation Analysis Between the Growth of Academic Achievement and CPSES Scores of the Students.

			Pre-Post Scores	***CPSES***
Pre-Post	Academic Success	Pearson Correlation	-.036	
		Sig. (2-tailed)	.841	
		N	34	

In Table 7, it is seen that there is a negative and low-level significant relationship between the growth of academic achievement and *CPSES* scores of the participants ($r = -0.036$; $p < .05$). Accordingly, it can be said that CPSES scores decrease when academic scores increase.

3.2. Interactive Video Monitoring and Academic Achievement

Interactive video monitoring rates were obtained in hours/seconds through IVS. In this context, the grouping is done as shown in Table 8. The study was carried out with 48 student data (N=48), as displayed in Table 8, by subtracting 12 students who did not complete one or both of the academic achievement tests and missing ones and did not write their names.

Table 8. Distribution of Students Completing Academic Achievement Tests According to Interactive Video Monitoring Groups

		Tag	**N**
Group No	0	Never viewed	17
	1	Viewed	14
	2	Actively Viewed	17
		Total	48

Since the number of students to be used in the analysis is less than 50, the Shapiro-Wilks test was used for normality analysis (Büyüköztürk, 2014). The normality distribution of achievement test data shows that parametric analysis methods can be used in data analysis. A Single Factor Covariance Analysis (One Factor ANCOVA) was used to analyze whether there was a significant correlation between interactive video monitoring rates (independent variable) and achievement test final application scores (dependent variable). Achievement test pre-application scores were used as the covariate. Explanations of the covariance analysis results are displayed in the table below.

Table 9. Results of Covariance Analysis of Final Application Scores

Source	**Sum of Squares**	**Degree of Freedom**	**Mean of Squares**	**F**	**p**
Achievement Test Pre-Application	44.289	1	44.289	5.184	.028
Group	35.596	2	17.798	2.083	.137
Bug	7099.000	44	34.941		

Table 9 shows that there was no significant difference in achievement scores compared to interactive video monitoring ratios (groups), $F(2, 44) = 2.083$, $p > .05$. In other words, academic success has no relation to interactive video viewing rates.

3.3. Interactive Video Monitoring and Learning Styles

In the second sub-problem of the study, it was investigated whether the learning style preferences differ significantly according to the interactive video monitoring rates. Learning style preferences were collected through the "Felder and Soloman Learning Styles Index." The results of this index show that each student has a learning style preference of 4 different factors and 2 different poles in each factor (Felder & Silverman, 1988). Each pole has three different levels: strong, medium and weak (Felder & Silverman, 1988).

The study was carried out with 46 students' data (N=46) by eliminating 14 students who did not fill out the learning styles index, fill out missing or write down their names. The non-parametric test, Kruskal Wallis(KW) H-test (Büyüköztürk, 2014), was used because the video monitoring rates did not show normal distribution. As shown in Table 10, According to the results of this study, interactive video monitoring rates (IVMR) did not differ according to learning style

preferences (p>.05), in other words, it was found that learning style preferences do not have different effects on interactive video monitoring (viewing) rates.

Table 10. Comparison Learning Style Preferences with the IVMR "Kruskal Wallis" Tests

Learning Style Factor	**N**	**Mean Rank**	**sd**	x^2	**p**
Think–strong and do– weak	19	20.92	2	1.230	.541
Do–strong and think– weak	14	25.39			
Medium	13	25.23			
Intuitive–strong and feeling– weak	19	22.37	2	.331	.848
Feeling-strong and intuitive– weak	8	25.50			
Medium	19	23.79			
Auditory–strong and visual– weak	10	26.40	2	.708	.702
Visual-strong and auditory – weak	17	23.41			
Medium	9	22.05			
Holistic – strong and sequential – weak	14	26.36	2	1.284	.526
Sequential – strong and holistic – weak	16	20.88			
Medium	16	23.63			

It was observed that students with SEQUENTIAL – Strong and HOLISTIC – Weak) learning style preference had the lowest video-monitoring average (Mean Rank: 20.88). Additionally, AUDITORY–Strong and VISUAL– Weak learning style preference had the highest video-monitoring average (Mean Rank: 26.40).

4. Discussion and Conclusion

In this study, the effects of interactive videos on students' academic achievement and self-efficacy perceptions on programming were investigated in the third-grade students of the Department of Computer Education and Instructional Technology. As the main result, it was found that interactive videos are not a significant variance in students' academic achievement and self-efficacy perceptions in programming education. Most of the studies in the literature, On the contrary of this research, has shown the positive effects of interactive video content on education (Duffy, 2008; Clothier, 2013; Altınpulluk, Kılınç, Mehmet & Onur, 2020). The underlying causes of this finding can be explained with quotations from the literature. Ronchetti

(2010) found that students thought that watching videos was tedious and burdensome. In this research, even in interactive content, students may have felt similar emotions. The size of videos can be another reason for the finding. Altınpulluk, etc. (2020), emphasized the segmentation and flexibility of video content and suggested to designers to divide the educational videos into "meaningful" parts to reach effective learning. If the case in the point is programming education, then the "meaningful" part is much more important, it can be suggested to researchers to think on this point meticulously. Similarly, Afify (2020), points out the length of interactive videos. He found that students watching short videos were more successful than others working with medium and long videos.

Secondly, it was found that students' learning style preferences did not have any impact on interactive video monitoring rates. In the literature, some of the studies support this finding. Guido and O'Connell(2015), investigated the link between learning style and online content usage in their studies. They found that the learning style of the individuals cannot be used as predictive data of online learning measures. Besides, Allert (2004) found no correlation between learning style and performance. As the result, it can be mentioned, even the content is programming language education, there is no relationship between learning styles and video monitoring rates of the students.

Additionally, the correlation between the variance in students' academic achievement and the variance in self-efficacy perception scores related to programming has been examined. It has been concluded that as academic achievement increases, the perception of self-efficacy about programming has decreased. This finding is opposite to the findings of Altun and Mazman (2012). Likewise, Tsai, Wang and Tsue (2019), have found a positive relationship between the programming experience and programming self-efficacy level of the students. Different results of the studies can be sourced from the characteristics of the research groups. The participants of this research were teacher candidates. Korkmaz and Altun (2014), has found that there was a significant difference between the programming self-efficacy perception scores of computer engineering and electrical-electronics engineering students. Therefore, it is believed that the change in the education process of students from different departments can be different.
In this research, the education of a script-based authoring tool was supported through interactive videos. The interaction of the videos was provided by using questions embedded in the video stream. For future research, the effects of different interaction technics in video-content to different programming courses can be investigated.

References

Afify, M. K. (2020). Effect of Interactive Video Length within E-Learning Environments On Cognitive Load, Cognitive Achievement and Retention of Learning. Turkish Online Journal of Distance Education, 21(4), 68-89.

Allert, J. (2004). Learning style and factors contributing to success in an introductory computer science course. Advanced Learning Technologies (s. 385-389). Joensuu, Finland: IEEE International Conference.

Akar, E. (2010). Yeni Eğitim Paradigması Olarak E-Öğrenme 2.0 ve Satış Elemanlarının Eğitiminde Kullanımı. Ç.Ü. Sosyal Bilimler Enstitüsü Dergisi, 19(1), s. 45-61.

Altınpulluk, H., Kılınç, H., Fırat, M., & Yumurtacı, O. (2019). The influence of segmented and complete educational videos on the cognitive load, satisfaction, engagement, and academic achievement levels of learners. Journal of Computers in Education, 1-28.

Altun, A., & Mazman, S. G. (2012). Programlamaya İlişkin Öz Yeterlilik Algısı Ölçeğinin Türkçe Formunun Geçerlilik ve Güvenirlik Çalışması. Eğitimde ve Psikolojide Ölçme ve Değerlendirme Dergisi, 3(2), s. 297-308.

Arslan, B., & Aksu, M. (2006). Orta Doğu Teknik Üniversitesi Mühendislik Öğrencilerinin Öğrenme Stili Profillerine Yönelik Betimsel Bir Çalışma. Eğitim ve Bilim, 31(141), s. 83-91.

Aşkar, P., & Davenport, D. (2009). An Investigation of Factors Related to Self-Efficacy for Java Programming Among Engineering Students. The turkish Online Journal of Educational Technology, 8(1).

Barthel, R., Ainsworth, S., & Sharples, M. (2013). Collaborative knowledge building with shared video representations. International Journal of Human-Computer Studies, 71(1), 59-75.

Brame, C. J. (2016). Effective Educational Videos. Retrieved from https://cft.vanderbilt.edu/guides-sub-pages/effective-educational-videos/

Bustamante, C. (2017). TPACK and teachers of Spanish: Development of a theorybased joint display in a mixed methods research case study. Journal of Mixed Methods Research, 13(2), 163-178.

Büyüköztürk, Ş. (2014). Sosyal Bilimler İçin Veri Analizi El Kitabı. Ankara: Pegem Akademi.

Büyüköztürk, Ş., Çakmak, E. K., Akgün, Ö., Karadeniz, Ş., & Demirel, F. (2010). Bilimsel Araştırma Yöntemleri. Ankara: Pegem Akademi Yayıncılık.

Clothier, P. (2013). Interactive Video: the next big thing in mobile. Retrieved May, 4, 2020.

Duffy, P. (2008). Engaging the YouTube Google-Eyed Generation: Strategies for Using Web 2.0 in Teaching and Learning. The Electronic Jouyrnal of e-Learning, 6(2), s. 119-130.

Ersoy, H., Madran, R. O., & Gülbahar, Y. (2011). Programlama Dilleri Öğretimine Bir Model Önerisi: Robot Programlama. Akademik Bilişim'11 - XIII. Akademik Bilişim Konferansı Bildirileri, (s. 731-736). Malatya.

Felder, R. M., & Henriques, E. R. (1995). Learning and Teaching Styles In Foreign and Second Language Education. Foreign Language Annals, 28(1), s. 21-31.

Felder, R. M., & Silverman, L. K. (1988). Learning and Teaching Styles in Engineering Education. Engineering Education, 78(7), s. 674-681.

Felder, R. M., & Soloman, B. A. (1994). Index of Learning Styles (ILS). Retrieved from http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSpage.html

Fraenkel, J., Wallen, N., & Hyun, H. (2011). How to Design and Evaluate Research in Education (8th b.). New York: McGraw-Hill Education.

Gözüm, S., & Aksayan, S. (1999). Öz-Etkililik-Yeterlik Ölçeği'nin Türkçe Formunun Güvenirliik ve Geçerliliği. Atatürk Üniv. Hemşirelik Yüksekokul Dergisi, 2(1), s. 21-34.

Howard, R. A., Carver, C. A., & Lane, W. D. (1996). Felder's Learning Styles, Bloom's Taxonomy, and The Kolb Learning Cycle: Tying It All Together In The CS2 Course. SIGCSE '96 Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education, 28, s. 227-231. New York, NY, USA.

Hrastinski, S., & Monstad, T. (2014). Exploring the relationship between the use of an interactive video website and organizational learning. New Media & Society, 16(4), 594-614.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. Computers in Human Behavior, s. 200-210.

Karaman, S., Yıldırım, S., & Kaban, A. (2008). Öğrenme 2.0 Yaygınlaşıyor: Web 2.0 Uygulamalarının Eğitimde Kullanımına İlişkin Araştırmalar ve Sonuçları. XIII. Türkiye'de İnternet Konferansı Bildirileri, (s. 35-40). Ankara.

Kolowich, L. (2016). 31 Video Marketing Statistics to Inform Your Strategy Retrieved from http://blog.hubspot.com/marketing/video-marketing-statistics#sm. 000qy2oh4113adblqf82 ndjr4muir

Korkmaz, Ö., & Altun, H. (2014). Adapting Computer Programming Self-Efficacy Scale and Engineering Students' Self-Efficacy Perceptions. Online Submission, 1(1), 20-31.

Kozma, R. B. (1986). Implications of instructional psychology for the design of educational television. Educational Technology Research & Development, 34(1), s. 11-19.

Kukul, V., & Gökçearslan, Ş. (2014). Scratch ile Programlama Eğitimi Alan Öğrencilerin Problem Çözme Becerilerinin İncelenmesi. 8th International Computer & Instructional Technologies Symposium, (s. 58-63). Edirne.

Litzinger, T. A., Lee, S. H., Wise, J. C., & Felder, R. M. (2005). A study of the reliability and validity of the Felder-Soloman Index of Learning Styles. Proceedings of the 2005 American Society for Education Annual Conference & Exposition, (s. 1-16).

Meij, H. v., & Meij, J. v. (2014). A comparison of paper-based and video tutorials for software learning. Computers & Education, 78, s. 150-159.

Munoz, C. L., & Towner, T. L. (2009). Opening Facebook: How to Use Facebook in the College Classroom. Proceedings of society for information technology & teacher education international conference. South Carolina.

Petan, A. S., Petan, L., & Vasiu, R. (2014). Interactive Video in Knowledge Management: Implications for Organizational Leadership. Challenges and Innovations in Management and Leadership – 12th International Symposium in Management, 124, s. 478-485.

Ramalingam, V., & Wiedenbeck, S. (1998). Development and Validation of Scores on a Computer Programming Self-Efficacy Scale and Group Analyses of Novice Programmer Self-Efficacy. Journal of Educational Computing Research, 19(4), s. 367-381.

Ronchetti, M. (2010). Using Video Lectures to Make Teaching More. International Journal of Emerging Technologies in Learning (iJET), 5(2), 45-48.

Samancı, N. K., & Keskin, Ö. M. (2007). Felder ve Soloman Öğrenme Stili İndeksi: Türkçeye Uyarlanması ve Geçerlik-Güvenirlik Çalışması. Ahi Evran Üniversitesi Kırşehir Eğitim Fakültesi Dergisi, 8(2), s. 37-54.

Schaffer, L. C., & Hannafin, M. J. (1986). The effects of progressive interactivity on learning from interactive video. ECTJ, 34(2), s. 89-96.

Seferoğlu, S., & Akbıyık, C. (2005). İlköğretim Öğretmenlerinin Bilgisayara Yönelik Öz-Yeterlik Algıları Üzerine Bir Çalışma. Eğitim Araştırmaları-Eurasian Journal of Educational Research, 19, s. 89-101.

Sönmez, V., & Alacapınar, F. G. (2013). Örneklendirilmiş Bilimsel Araştırma Yöntemleri. Ankara: Anı Yayıncılık.

Thomas, L., Ratcliffe, M., Woodbury, J., & Jarman, E. (2002). Learning styles and performance in the introductory programming sequence. SIGCSE '02 Proceedings of the 33rd SIGCSE technical symposium on Computer science education, 34, s. 33-37. New York, NY, USA.

Tsai, M. J., Wang, C. Y., & Hsu, P. F. (2019). Developing the computer programming self-efficacy scale for computer literacy education. Journal of Educational Computing Research, 56(8), 1345-1360.

Tu, C., Blocher, M., & Ntoruru, J. (2008). Integrate Web 2.0 technology to facilitate online professional community: EMI special editing experiences. Educational Media International, 45(4), 335–341.

Yıldırım, N., & Özmen, B. (2012). Video Paylaşım Sitelerinin Eğitsel Amaçlı Kullanımı. e-Journal of New World Sciences Academy, 7(1).

Yin, Z.-X., Lin, C.-H., Yang, C.-T., & Chen, Z.-Z. (2013). An interactive-video system to medical e-learning. Orange Technologies (ICOT), (s. 321-324). Tainan.

Zhang, D., Zhou, L., Briggs, R. O., & Nunamaker Jr., J. F. (2006). Instructional video in e-learning: Assessing the impact of interactive video on learning effectiveness. Information & Management, 43(1), s. 15-27.