



Academic Achievement in Computer Programming Instruction and Effects of the Use of Visualization Tools; at the Elementary School Level

Işıl Gülmez¹ and Nesrin Özdener^{2*}

¹Şirinevler Middle School, Bursa, Turkey.

²Marmara University, Atatürk Faculty of Education, Department of Computer and Instructional Technologies, Istanbul, Turkey.

Authors' contributions

This work was carried out in collaboration between both authors. The authors worked together to design the study and write the protocol. Author NÖ supervised the work. Author IG collected the data, performed statistical analysis. Both authors worked on the construction of the manuscript through the points of initial drafts to the final approved manuscript. Both authors read and approved the final manuscript.

Article Information

DOI: 10.9734/BJESBS/2015/18316

Editor(s):

(1) Vlasta Hus, Department of Elementary Teacher Education, University of Maribor, Slovenia.

Reviewers:

(1) Anonymous, Chulalongkorn University, Thailand.

(2) Nancy Maynes, Nipissing University, Canada.

(3) Barry Chametzky, Ozarks Technical Community College, USA.

Complete Peer review History: <http://sciencedomain.org/review-history/10248>

Original Research Article

Received 15th April 2015

Accepted 30th June 2015

Published 19th July 2015

ABSTRACT

This study investigated the effects of using visualization tools in programming instruction on student success and motivation. The study also sought to determine courses that significantly correlate with elementary school students' programming success. Two experimental groups participated in the study. One group used a flowchart model tool; the other used a narrative tool. The groups' success at using variables, conditions and loops (a programming function that iterates a statement or condition based on specified boundaries) were compared. Results revealed that there was no difference on using variables but that there were significant differences between using conditions and loops and using narrative tools. There was no difference between the groups' motivation. The

*Corresponding author: E-mail: nozdener@marmara.edu.tr;

courses which significantly correlate with students' algorithm development success are Turkish, math, English, and information technologies.

Keywords: Visualization tools; programming instruction; academic achievement.

1. INTRODUCTION

Programming courses have become prevalent because they are important parts of computer literacy as well as contributors to professional competence; it is known that these types of courses are useful for the development of students' analytical thinking and problem solving skills [1,2]. Similarly, the importance of algorithmic thinking is emphasized in the ACM (Association for Computing Machinery) curriculum, developed for K-12 computer sciences courses [3]. For these reasons, introductory programming courses are added to the primary and elementary education curriculums in various countries [4]. A software program titled Lego Mindstorms, providing programming in order to effectuate the required commands by a robot in programming courses, is being used in schools in Taiwan [3] and in Korea, where the decision was made to carry out changes in informatics curriculums of elementary schools in 2010 and high schools in 2011. According to recent studies, it is necessary to carry out these changes in order to enable primary school students to understand the algorithms [5]. In India, the programming education starts with the teaching of basic conceptions by the use of the LOGO programming language at the 3rd class level, and subsequently, the aim is for the students to draw flow diagrams step-by-step and learn to write programs at a basic level with the BASIC language until the end of the 8th class [6]. In Israel and Canada, the courses with relation to the programming teaching are given in high schools [4].

In Turkey, the aim of teaching students skills in developing algorithms and allowing them to find suitable solutions to problems is stated among the objectives of the course of information technologies for elementary education. According to Arabacıoğlu, Bülbül & Filiz [7], the programming education in Turkey is given with theoretical methods; however, the literature shows that the theoretical methods are not effective [8,9]. On the other hand, according to the research it is important to give examples that can teach the programming logic and solving problems with computational thinking, instead of

teaching certain programming language codes [10-14].

Many studies, and the experiences of a large number of teachers, agree that learning to program is not easy [15,16]. To improve the teaching and learning environment, development of appropriate learning aids is therefore highly relevant [16]. In recent years, many tools have been proposed to reduce the programming learning difficulties felt by many students [17,18]. For this purpose visualization tools are designed in order to develop students' algorithm development skills. According to Cooper et al. [19], visualization tools are novice programming environments, providing a visual interface in which programs are constructed using simple gestures, such as drag and drop. Such visualizations can serve as conceptual models that help learners construct viable programming knowledge [18]. Moreover, the research emphasizes the effect of visibility and interaction, used in the software, on student success and motivation [7,8,11,16,18,20-23].

The researchers have laid out classification of software visualization tools for use in introductory programming education and explained their characteristics [18,24,25]. There are various opinions about which of these tools should be used at the initial stage of the programming course. Some of the researchers claim that the tools having a flow chart model are appropriate for the beginners [17,26,27]. Some others, however, argue that narrative tools are more convenient in this respect [12,28]. Bravo et al. [29], indicated that the tools helping the formation of algorithm by means of a character may facilitate students' comprehension and analysis of the program. A previous study by Ramadhan [14], carried out with university students, indicated that the support of visual tools has a positive effect on student success in relation to condition and loop. Rößling [15], stated that the approaches which allow placement of objects make understanding the dynamic behavior of algorithms and data structures easier. The researchers found out that this approach increases student motivation. According to the research results of Lahtinen, Ahoniemi and Salo [30] the necessity of visualization tools increases as the complexity of problems increases.

A comparative analysis shows improvement in student performance when using image processing. Shesh [31], claims that the image processing approach can be fit to courses for other majors where programming is considered useful but not critical. Burneo et al. [32] claim that it is possible to foster a degree of basic skills acquisition and stimulate creativity in grade school students by incorporating Scratch. Hyeonjin et al. [1], examined the new roles of teachers for efficiency and innovation and stated that new approaches, focusing on thinking skills rather than technical skills, helped teachers to develop adaptive expertise.

Leiva and Salas [33] claim that using a game-based approach helped to foster collaboration, participation, motivation and student creativity. Therefore, using Scratch would be useful for teaching programming to novice students. Results revealed that these tools, if effectively used, can improve students' alertness and interest in the subject, and ultimately produce positive results [34]. According to Kalogeropoulos et al. [35], the actual programming abilities of the students are better evaluated using computer-based assessment tests. Using visualization tools such as Scratch should be an appropriate way of evaluating programming abilities. Zeeman [36], states that implementation of Scratch at the first-year level could serve to bridge the gap between the schooling system and university in terms of the development of critical thinking-skills.

However, the studies conducted on this subject are performed at university level and remain deficient in determining whether the visualization tool would conform to the needs of elementary education students. Therefore, it is important to examine required ways and methods by which the programming education can be conducted for elementary school students. Under these circumstances, it is important to discover which visualization tools will benefit elementary school students who are learning the basics of programming.

Another point where the research remains insufficient is that the studies, made with relation to programming education, are not provided in consideration of the student level. Likewise, because matters related to programming have been newly added to the elementary education curriculum in various countries worldwide, the number of examples that can be studied according to the ages and levels of students is

limited [3,4,5]. In this context, it is important to emphasize the selection of a visualization tool suitable for elementary education on the one hand, and on the other develop the methods needed to enhance student success and motivation.

Getting to know the factors affecting the success of students in programming will help educators or instructors to orient the students towards related departments. According to the research, there is a positive correlation between students' academic success generally and their success in a university programming course [37]. In various studies, significant positive correlations were found between the students' programming success and mathematics achievement [10,38-41]. Furthermore, research has shown that problems in learning programming concepts are correlated with problem solving skills and the first language of the student [15]. Some studies showed a positive significant relationship between the success of the students in their English classes and in programming [42,43]. However, the research results of Jones and Burnett [44] found no correlation between English or foreign language success and programming success. The research is far from being sufficient to determine the effect of academic success of the students, especially those starting with programming, on their programming success. Therefore, there is need to research whether for elementary school students who haven't taken a programming course before their success in information technologies, Turkish, math and English, predicts their programming achievement.

1.1 Purpose

The main purpose of the study is to compare narrative tools with flowchart model tools, which are both used in the programming education given within the information technologies course, in terms of student success and motivation. The research also aims to determine the effect of academic success on success in developing algorithms. The following hypotheses are tested within the scope of the research.

1. When the students using flowchart model tools are compared with the students using narrative tools, in terms of their success using variables, there will be a significant difference in favor of the students using flowchart model tools.

2. When the students using flowchart model tools are compared with the students using narrative tools, in terms of their success using condition and loop, there will be a significant difference in favor of the students using narrative tools.
3. When the students using flowchart model tools are compared with the students using narrative tools in terms of their motivation, there will be a significant difference in favor of the students using narrative tools.
4. There is a meaningful positive correlation between the academic success of the students in their information technologies, math, Turkish, and English courses and their success in developing algorithms.
5. The academic achievements of the students in their information technologies, math, Turkish (L1), and English (L2) courses predict their achievement in algorithm development in a significant manner.

2. MATERIALS AND METHODS

The study, which used an experimental and correlative survey model, was conducted with two experimental groups. In the research, a post-test experimental model was used for comparing the effect of the visualization tools on algorithm development success of the students, and a pre-test post-test experimental model was used for determining the effects of visualization tools on their motivation.

In the study, a correlative survey model was used for determining the relation between the algorithm development success and academic success of the students. The students who took part in the study groups were distributed by a random method. One of the groups (Experimental Group 1) used the flowchart model tool and the other (Experimental Group 2) used the narrative tool. The student groups were exposed to the tools for eight weeks. The study was done in a computer laboratory and was administrated by the researcher. Relevant applications pertaining to the value assignment, loop, condition, and basic conceptions were made by having the experimental groups use different visualization tools. An example of such applications is described in Appendix A. The students who participated in the study group were distributed by a random method. A pre-knowledge test (Appendix B) was given in order to determine whether the groups were equal in their algorithm development success.

At the end of applications, the results of two post-tests (Appendix C, Appendix D) were used in order to compare the effects of the two visualization tools on algorithm development success. In order to increase the reliability of results, paper-based (Appendix C) and the computer-based test (Appendix D) were applied and the results of these tests were used. Thus, researchers tried to determine whether the results of these two examinations were affecting each other. Furthermore, in order to determine the effect of the visualization tools on the motivation of students, the motivation test developed by Özerbaş [45], was applied twice, before and after the application, and then results were compared with each other.

2.1 Research Population

The study group of the research consisted of 84 students who had taken programming education in elementary school. Participants in this study were elementary school students ranging in age from 12 to 14. The students were novice programmers. The students who participated in the study group were distributed by a random method. The study groups were distributed as shown Table 1.

2.2 Research Instruments

For the study, data were collected on the pre-knowledge test, motivation measurement, two post-test exams, and the average of students' grade points. A list containing the targeted behaviors was prepared in order to ensure the content validity of the pre-knowledge test, and post-tests; then four separate computer teachers were required to evaluate whether questions measured these target behaviors. For this purpose, a list containing the targeted behaviors to be measured by the questions was given to the teachers, and each of the teachers was required to write the question number next to the target behaviors. All tests used in this research were applied in students' native language, then translated into English in order to use in this article.

2.2.1 Pre-knowledge test

The pre-knowledge test, developed by the researcher (Appendix B), was used in order to determine whether the groups that used different visualization tools were equal in terms of algorithm development success. At the end of the validity study of the scope of pre-knowledge test, consistency between results of the

evaluation carried out by four teachers was determined to be 87.5%. In order to estimate the reliability of the questions given in the test, a further consistency reliability analysis was conducted, based on Cronbach's Alpha test, using SPSS. The alpha coefficient showed the internal consistency to be 0.73.

2.2.2 Motivation scale

The motivation scale, developed by Özerbaş [45] was applied in order to determine the motivation for learning. The entire scale consists of 30 Likert-type expressions, 15 positives and 15 negatives. Each item in the measurement scale was assessed by a five-point Likert scale. A total greater than 90 points in the results, is a sign of positive behaviors and points under this number are sign of negative behaviors. The validity and reliability study of this motivation scale was carried out by Özerbaş [45]. Factor analysis was used for the structure validity, and Cronbach's Alpha reliability value of the tool was calculated as 0.88.

2.2.3 Paper-based test (Post-Test)

The researchers developed the post-test to assess the success of students' algorithm development at the end of application (Appendix C). The examination on paper consisted of 16 questions and was evaluated based on a maximum score of 100. At the end of the validity study of the scope of the paper-based test, consistency between results of the evaluation carried out by four teachers was determined to be 93.75 %. The internal consistency coefficient of the test (Cronbach's alpha) was calculated to be 0.80.

2.2.4 Computer-based test (Post-Test)

The Computer-based test (application exam) was used to evaluate the success of students in algorithm development after the completion of application (Appendix D). Five questions were asked in the exam, held at the computer lab, and the exam was evaluated based on a maximum of 100 points. The application exam was held in order to increase the reliability of the study. At the end of the validity study of the scope of the

computer-based test, consistency between results of the evaluation carried out by four teachers was determined to be 90%. The internal consistency coefficient of the test (Cronbach's alpha) was calculated to be 0.76.

2.2.5 Grade point average (GPA) of students

The data related to the grade point average of the students participating in the study groups were acquired from school entries. The GPAs (Grade Points Average) were those of their courses that were completed in the previous period.

2.3 Material Used

Cooper et al. [19] defined narrative tools as environments of interactive animation, which are used for introducing novices to programming. According to research, flow model tools construct programs by connecting program elements in a fashion that represents the order of computation and specialized output realizations. In this research, the applications were carried out using Fcpro and Scratch visualization tools. During the study, the activities (Appendix A), carried out using both of these tools were prepared and applied by the researchers.

2.3.1 Scratch

The Scratch tool used during the research is a narrative tool for making a story out of an algorithm, developed in order to teach the programming to the students. With the help of Scratch, the students are able to take the feedbacks of the codes that they created by the drag-and-drop method in a visual manner. The display (screen) image of the Scratch software is shown in Fig. 1.

2.3.2 Fcpro

The Fcpro tool that is used in the research is a flow chart model tool ensuring the creation of an algorithm by bringing the codes through the drag-and-drop method. The screen image of the Fcpro tool is shown in Fig. 2.

Table 1. Numbers and percentage distributions of the students of the study groups

Groups	Tools	Number of students	Percentage %	Sex	
				Female	Male
Exp. Group 1	Flow chart model	42	50	21	21
Exp. Group 2	Narrative	42	50	21	21

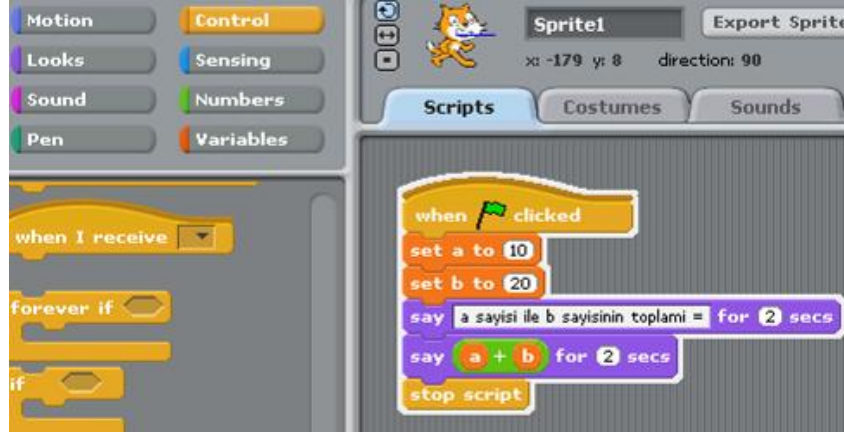


Fig. 1. Screen image of the scratch software



Fig. 2. Screen image of the Fcpro tool

3. FINDINGS

Before beginning this research, the question whether the students constitute equal groups both in terms of algorithm development success and motivation was considered. The following findings were obtained.

3.1 Pre-Application Findings

According to the result of the Shapiro-Wilk test, made in order to determine whether the study groups were equal in algorithm development success, the groups displayed a normal distribution ($p > .05$). The pre-knowledge test points were compared through an independent sample t-test; the results of which are shown in Table 2. The pre-knowledge test showed no

significant differences between the group successes [$t(82) = .33, p > .05$].

At the beginning of the study, the groups which showed a normal distribution with respect to algorithm development success were tested in order to determine whether the groups were also equal in terms of motivation. According to the result of the Shapiro-Wilk test, the groups displayed a normal distribution ($p > .05$). The pre-test results of the groups have been compared through independent sample t-test. The results are shown in Table 3.

There was not a significant difference between the groups' motivation pre-test results [$t(82) = 1.50, p > .05$] at the end of the independent sample t-test. In this case, it is possible to say

that the study groups were equal in both motivation and pre-knowledge.

3.2 Post-Application Findings

Hypothesis 1: When the students using flowchart model tools are compared with the students using narrative tools, in terms of their success on the use of variables, there will be a significant difference in favor of the students using flowchart model tools.

According to the result of the Shapiro-Wilk test, applied to questions in the post-test that pertaining to the variables the exam results of the students using both visualization tools show normal distribution ($p > .05$). The results of the independent sample t-test, also applied to the questions in the post-test that pertained to the variables are shown in Table 4. According to these results, there was no a significant difference on the use of variables between the groups [$t(82) = 1.65, p > .05$], [$t(82) = 0.71, p > .05$].

Hypothesis 2: When the students using flowchart model tools are compared with the students using narrative tools, in terms of their successes on the use of condition and loop, there will be a significant difference in favor of the students using narrative tools.

According to the result of the Shapiro-Wilk test, applied to the questions on the post-test that pertained to the condition and loop expressions, the exam results of the students using both visualization tools show normal distribution ($p > .05$).

At the end of the research, the results of the independent sample t-test pertaining to the both paper-based and computer-based examination questions, which were made in order to increase the reliability of the post test results, are shown in Table 5. Table 5 shows that according to the

independent sample t-test results there was a significant difference in favor of the students using narrative tools on the use of both condition and loop expressions ([$t(82) = 2.57, p < .05$], [$t(82) = 2.02, p < .05$], [$t(82) = 4.17, p < .05$], [$t(82) = 7.09, p < .05$].

Hypothesis 3: When the students using flowchart model tools are compared with the students using narrative tools in terms of motivation, there will be a significant difference in favor of the students using narrative tools.

3.3 Comparison of the Groups' Motivation Pre-Test and Post-Test Points

For the comparison of the pre-test and post-test points for the two groups on the motivation preliminary test, the Shapiro-Wilk test was given to ensure that the motivation post-test results displayed normal distribution, which they did ($p > .05$). A dependent samples' t-test was used to determine the correlation between the motivation preliminary test points and motivation post-test points of the students. The results of the dependent samples' t-test, are shown in Table 6.

When the results of the motivation pre-test and post-test were compared through dependent samples' t-test, the result showed a significant difference in favor of the post test results [$t(83) = 2.54, p < .05$].

3.4 Comparison of the Motivation Post-Test Points of the Groups

An independent sample t-test was applied in order to compare the motivation post-test results of the groups. The results are shown in Table 7. The independent sample t-test determined that there was no significant difference between the groups [$t_{(82)} = .455, p > .05$].

Table 2. T-test results relating to the pre-knowledge test scores

Gender	N	X	SS	Sd	T	p
Female	42	34.83	22.71	82	.33	.74
Male	42	36.36	18.93			

Table 3. Independent sample t-test results relating to the motivation pre-test results

Tools	N	X	SS	Sd	T	p
Flowchart model	42	103.09	12.97	82	1.50	.14
Narrative	42	98.33	16.01			

* $p < .05$

Table 4. The results of the independent sample t-test relating to the post-tests

Post-test	Tools	N	X	SS	Sd	t	p
Paper-based	Flowchart model	42	7.40	3.55	82	1.65	.10
	Narrative	42	8.54	2.76			
Computer-based	Flowchart model	42	45.04	7.86	82	0.71	.47
	Narrative	42	46.26	7.55			

*p<.05

Table 5. The results of the independent sample t-test relating to the to the post-tests

Post-test		Tools	N	X	SS	Sd	t	p
Paper-based	Condition	Flowchart model	42	12.54	7.21	82	2.57	.01
		Narrative	42	16.28	6.06			
	Loop	Flowchart model	42	11.14	5.26	82	2.02	.04
		Narrative	42	13.40	4.97			
Computer-based	Condition	Flowchart model	42	13.07	3.40	82	4.16	.00
		Narrative	42	15.85	2.68			
	Loop	Flowchart model	42	11.61	3.29	82	7.09	.00
		Narrative	42	16.19	2.58			

*p < .05

Table 6. The results of the paired sample t-test for the comparison of the motivation pre-test and post-test points of the groups

Motivation	N	X	S	Sd	t	p
Pre-test	84	100.71	14.68	83	2.54	.01
Post-test	84	105.35	15.76			

*p < .05

Table 7. The results of the independent sample t-test relating to the motivation post-test results of the groups

Tools	N	X	S	Sd	t	p
Flowchart Model	42	106.14	16.41	82	.45	.65
Narrative	42	104.57	15.24			

*p < .05

Hypothesis 4: There is a meaningful correlation in a positive direction between the academic success of the students in their information technologies, math, Turkish and English courses and their success in algorithm development.

The grade point averages (GPAs) scored by the students in their post-test examinations, are used to determine which courses have a correlation with the programming achievement of the students in this part of the research. First of all, the Kolmogorov-Smirnov test was used to determine whether the exam results and course grades displayed normal distribution. The test showed that post-test averages and GPAs of the information technologies, math, Turkish and English courses of the students displayed normal distribution ($p > .05$). A Pearson correlation test

was used to determine the correlation between the students' academic achievements in the courses of information technologies, math, Turkish and English and post-test point averages in algorithm development. The results of the correlation test, made in this respect, are shown in Table 8.

Table 8 shows that there is a significant correlation between the algorithm development post-test averages and students' success in information technologies, math, Turkish and English courses ($p < .050$). In the table it appears that there is a positive significant correlation at relatively high level between the post-test averages and academic achievements in the English course, following the correlation test for the academic achievements and post-test results ($r = .74, p < .05$). Furthermore, it is also detected

that there is a positive significant correlation at a medium level between post-test results and the academic achievements of the students in the courses of Turkish ($r = .66, p < .05$), math ($r = .60, p < .05$), and information technologies ($r = .55, p < .05$).

Table 8. The results of the Pearson correlation test for independent variables and algorithm development post-test average

Algorithm development post test average		
	R	P
Information technologies	.55	.00
Turkish	.66	.00
Math	.60	.00
English	.74	.00

Hypothesis 5: The academic achievements of the students in their information technologies, math, Turkish and English courses predict their algorithm development achievements in a significant manner.

Multi-regression analysis was conducted in order to determine variables predicting the programming achievement. The results of this analysis are shown in Table 9.

Table 9. Variables predicting the programming achievements

Variables	B	T	P
Constant	21.57	4.30	.00
Information technologies	.16	1.51	.13
Turkish	.16	1.68	.10
Math	-.02	-.22	.82
English	.34	3.84	.00

$$R = 0.76, R^2 = .57; F = 26.77, p = 0.00$$

The achievements in the information technologies, math, Turkish and English courses, which were hypothesized to predict the algorithm development achievement, have a positive significant correlation at a medium level with the algorithm development achievement ($R = 0.76, R^2 = 0.57, p < .01$). The results showed that the contribution of the predictor variables accounted for 57% of total variance over the algorithm development achievement. The order of predictive variables according to the standardized regression coefficient (β) is as follows: English (.34), Turkish (.16), information technologies (.16), and math (.02).

4. DISCUSSION

In this study, the effects of using visualization tools in programming instruction on student success and motivation were investigated. The study also sought to determine courses that significantly correlate with programming success.

At the end of the research, no significant difference was found between the achievements of the students using flowchart model tools and students using narrative tools on the use of variables. Although the studies examined in the literature indicate that the flowchart model is more useful in teaching basic conceptions at the beginning level of programming education [26, 27] these studies were designed for university students, a different target group which is different than elementary school students both in age group and preliminary knowledge. These differences may be the cause of the difference between the results of the present study and the results of the studies examined in the literature.

According to another outcome of the research, a significant difference is found in favor of the students using narrative tools on the condition and loop utilization. It is possible that the reason for this difference is that using animation to aid visualization of the condition and loop expressions can facilitate the comprehension and retention of these concepts. Bravo et al have emphasized this topic and indicated that tools that use a character to aid students in creating algorithms can help the students to understand and analyze programs [29]. This outcome, obtained in the research, supports the research that emphasizes the effect of visual elements and interaction in software on student achievement [7,8,11,14].

According to another outcome obtained from the research, there is no significant difference between the motivations of the students using narrative tools and those of students using flowchart model tools, whereas previous research claims that the use of graphics and animation in software is more effective in increasing the interest of the students in courses [11,14,21-23]. Therefore, it appears that the narrative tool increases the motivation of students more than the flow chart model tool. This may be because the tools that make a story out of the algorithm are more visual and more attention distracting for the elementary education students. However, the motivation post-test results at the end of the research did not show

the predicted difference. When the motivation pre-test and motivation post-test points are compared, it is observed that there is a significant difference in favor of the post-test results; as a consequence, the visualization tools used in this context increase the motivation of the students. This outcome may be interpreted to mean that the students have not conducted any activity in advance with these types of tools and the visualization tools may be more interesting for them. Perhaps if the visual tools are used for longer periods, they may increase the importance of visualization for students and increase their motivation.

Another outcome of the research shows that a significant positive correlation exists between the students' algorithm development achievements and their academic achievements in information technologies, math, Turkish, and English. This outcome supports the study on this subject, which indicate that students' experience using computers affects their achievements in programming [37]. When the studies carried out in this context are examined, it was expected that a correlation at a high level will exist between the achievements of students in the information technologies course and their achievements in algorithm development; however, this research resulted in no such relationship at the expected level. This outcome may be due to the content of information technologies courses, which may not have presented the concepts and definitions of programming in a comprehensive manner.

The outcome of the research showing that there is a significant correlation between math course achievements and programming achievements coincides with the outcomes of studies carried out at the related university level. These studies show a significant positive correlation between achievement in math and achievement in programming [38-41,46,47]. The outcome obtained in this research may be explained by the fact that the comprehension and interpretation of the questions requiring a simple mathematical process and skills in developing appropriate solution proposals to the questions have effects over the algorithm development. The outcome of the study indicating that there is a significant correlation between the achievements of the students in the Turkish (mother tongue) course and in programming may be explained by the fact that the Turkish course achievements contribute to the comprehension of questions pertaining to the algorithm development and their interpretations.

The outcome indicating that there is a significant correlation between the English (foreign language) achievements and the programming achievements of the students supports the research result of Nowaczyk [42]. The correlation between English and algorithm development achievement in the findings of this research may be explained by the fact that the commands for algorithm development are mostly in English, so students who do well in English can understand these commands more easily.

This study showed that the academic achievements of the students in math, information technologies, Turkish, and English courses predict algorithm development achievements in a significant manner. When the contribution of student academic achievements to their programming achievement is examined, the order of importance of these courses is as follows: English, Turkish, information technologies, and math. The English course takes the first place in the prediction of programming achievement, probably because high achievement in English helps the students comprehend the programming concepts and, as a consequence, facilitates their use of visualization tools.

5. CONCLUSION

In conclusion, the use of tools and methods that may address the elementary education students is important in programming education. The use of visualization tools in teaching programming increases student achievement and motivation. Under these circumstances, it will be useful for the institutions graduating teachers to train teacher candidates on the auxiliary tools and methods that may increase student success and motivation pertaining to algorithm development. Furthermore, the acquisition of knowledge on the courses requiring skill in algorithm development may cast light on determining what needs to be done to further develop this skill.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Hyeonjin K, Hyungshin C, Jeonghye H, Hyo-Jeong. S. Enhancing teachers' ICT capacity for the 21st century learning

- environment: Three cases of teacher education in Korea. *Australasian journal of educational technology*. 2012;28(6):965-982.
2. Sleeman D, Putnam RT, Baxter JA, Kuspa LA. Pascal and high-school students: a study of errors. *Journal of Educational Computing Research*. 1986;2(1):5-23.
 3. Lin J, MC Yen, Yang LY, Chen MC. C.-F. Teaching computer programming in elementary schools: A pilot study. *National educational computing conference, Philadelphia, PA. Pennsylvania; 2005*. Available:http://www.stagecast.com/pdf/research/Lin_NECC2005_Paper_RP.pdf (Accessed 11 March 2013)
 4. Tucker A, Deek F, Jones J, McCowan D, Stephenson C, Verno A. A Model curriculum for K12 computer science: final report of the Association for Computing Machinery (ACM), New York; 2003. Available:http://www.isp.org.pl/podstawa/podstawa_files/K12_Computer_Science.pdf (Accessed 25 December 2008)
 5. Choi SK, Bell T, Jun SJ, Lee WG. Designing offline computer science activities for the Korean elementary school curriculum. *Proceedings of the 13th annual conference on Innovation and technology in computer science education*. Madrid, Spain. 2008;338-338.
 6. Curriculum and teaching material for K-12 Indian schools Computer science; 2007. Available:<http://www.it.iitb.ac.in/~sri/papers/SSRVM-CS-March07.pdf> (Accessed 25 December 2008)
 7. Arabacıoğlu C, Bülbül H, Filiz A. A new approach to computer programming teaching. *Annual conference of academic informatics, Dumlupınar University, Turkey*. September, 4, 2007. Available:<http://ab.org.tr/ab07/bildiri/99.doc> (Accessed 3 March 2008)
 8. Gültekin K. The effects of multimedia on computer programming achievement. Master's thesis. Hacettepe University Institute of Graduate Studies in Science, Ankara; 2006.
 9. Garner S. Learning resources and tools to aid novices learn programming: Informing Science Information Technology Education Joint Conference. Finland. 2003;213-222.
 10. Özdenler N. A comparison of the misconceptions about the time-efficiency of algorithms by various profiles of computer programming students. *Computers & Education*. 2008;51:1094-1102.
 11. Kelleher C, Pausch R, Kiesler S. Storytelling Alice motivates middle school girls to learn computer programming, *proceedings of the SIGCHI conference on human factors in computing systems, San Jose, California, USA*. 2007;1455-1464. DOI:10.1145/1240624.1240844.
 12. Malan DJ, Leitner HH. Scratch for budding computer scientists. *ACM SIGCSE Bulletin*. 2007;39(1). DOI: 10.1145/1227504.1227388.
 13. Cooper S, Dann W, Pausch R. Teaching objects-first in introductory computer science. September 1, 20, 2007, *Proceedings of the 34th SIGCSE technical symposium on Computer science education, Reno, Nevada, USA*. 2003;191-195. DOI: 10.1145/611892.611966.
 14. Ramadhan HA. Programming by discovery. *Journal of Computer Assisted Learning*. 2000;16:83-94.
 15. Röfling G. A family of tools for supporting the learning of programming. *Algorithms* 2010;(3):168-182.
 16. Ansari MB. Incorporating visual and animation teaching tools in computer programming classes for effective teaching and learning. Master's. Faculty of Information Technology and Multimedia Communications Open University Malaysia. 2011.
 17. Santos A, Gomes A, Mendes AJ. Integrating new technologies and existing tools to promote. *Algorithms* 2010;(3):183-196.
 18. Sorya J, Karavirta V, Malmi L. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education*. 2013;13(4).
 19. Cooper S, Powers K, McNally M, Goldman KJ, Proulx V, Carlisle M. Tools for Teaching Introductory Programming: What Works? *37th SIGCSE Technical Symposium on Computer Science Education, 2006, (s.560-561)*. Houston, Texas, USA. Available:<http://dsys.cse.wustl.edu/resources/papers/gross-sigcse-2006.pdf> (Accessed 29 September 2010)
 20. Sondag T, Pokorny KL, Rajan H. Frances-A: a tool for architecture level program visualization. *Journal of Computing Sciences in Colleges*. 2011;26(5):283-292.
 21. Bishop-Clark C, Courte J, Howard EV. A quantitative and qualitative investigation of

- using Alice programming to improve confidence, enjoyment and achievement among non-majors. *Journal of Educational Computing Research*. 2007;37(2):193-207.
22. Brusilovsky P, Spring M. Adaptive, Engaging, and Explanatory Visualization in a C Programming Course. *Proceedings of ED-MEDIA'2004 - World conference on educational multimedia, Hypermedia and Telecommunications*. AACE, 1264-1271. Switzerland; 2004.
 23. Lin C, Zhang M. The use of computer animation in teaching discrete structures course. *MICS 2003. Proceedings the 36th Annual Midwest Instruction and Computing Symposium*; 2003.
Available:<http://www.micsymposium.org/apache2-default/mics2003/Lin.pdf>
(Accessed 26 March 2008)
 24. Powers K, Gross P, Cooper S, McNally M, Goldman KJ, Proulx V, Carlisle M. Tools for teaching introductory programming: What works? *Proceeding of the 37th SIGCSE technical symposium on computer science education*. Houston, Texas, USA. 2006;560- 561.
 25. Bergin J, Martinez MP. An overview of visualization: its use and design: Report of the Working Group on Visualization, Integrating Tech. Into C.S.E. *ACM SIGCSE Bulletin* 28(SI),1996;192-200.
 26. Baldwin LP, Kuljis J. Visualization techniques for learning and teaching programming. *Journal of Computing and Information Technology*. 2000;8(4):285–291.
 27. Crews T, Ziegler U. The flowchart interpreter for introductory programming course. *Frontiers in education conference*. Tempe, Arizona. 1998;307-312.
 28. Grissom S, McNally MF, Naps T. Algorithm visualization in CS education: comparing levels of student engagement, *Proceedings of the 2003 ACM symposium on Software visualization*. California. 2003;87-94:11-13.
DOI: 10.1145/774833.774846.
 29. Bravo C, Marcelino MJ, Games A, Esteves M, Mendes AJ. Integrating educational tools for collaborative computer programming learning. *Journal of Universal Computer Science*. 2005;11(9):1505-1517.
 30. Lahtinen E, Ahoniemi T, Salo A. Effectiveness of integrating program visualizations to a programming course. *Proceedings of the 7th Baltic sea conference on computing education research*. Koli National Park, Finland. 2007;195-198.
 31. Shesh A. High-level application development for non-computer science majors using image processing. *Computers & Graphics-UK* 2012;36(3): 170-177.
 32. Burneo L, Lucas R, Lopez P, et al. Generating new collaborative learning environments in primary education using Scratch. *5th International Conference of Education, Research and Innovation (ICERI)*, Madrid, Spain. 2012.
 33. Leiva AJF, Salas ACC. Practices of advanced programming: Tradition versus innovation. *Computer Applications in Engineering Education*. 2013;21(2):237-244.
 34. Stephen M, Franklin W, Elizabeth A, Juma K, Patrick O. Teaching computer programming in the 21st century. *International Journal of Science and Technology*. 2011;1(6):2224-3577.
 35. Kalogeropoulos N, Tzigounakis I, Pavlatou EA, Boudouvis AG. *Computer-Based Assessment of Student Performance in Programming Courses*. 2013;21(4):671-683.
 36. Zeeman M. To scratch or not to Scratch - a Reflection. *Proceedings of the 8th International conference on e-learning*, North West University, Vanderbulji Park, South Africa; 2013.
 37. Leeper RR, Silver JL. Predicting success in a first programming course. *13th SIGCSE Technical Symposium on Computer Science Education*. Indianapolis, Indiana, United States. 1982;147-150.
DOI:10.1145/800066.801357.
 38. Erdoğan B. An analysis of the correlations between programming success and general academic achievement, general aptitude, attitude toward computer, gender and high school type. Master's thesis. Marmara University Institute of Educational Sciences, Istanbul; 2005.
 39. Webb NM. Cognitive requirements of learning computer programming in group and individual settings. *AEDS Journal*. 1985;18(3):183-193.
 40. Fletcher SH. *Cognitive Abilities & Computer Programming*. 1984; EDRS (ED259700).
 41. Pea RD, Kurland DM. On the cognitive prerequisites of learning computer

- programming (Technical Report No.18). New York: Bank Street College of Education, Center for Children and Technology; 1983.
Available:http://hal.archives-ouvertes.fr/docs/00/19/05/31/PDF/A17_Pe_a_Kurland_83.pdf
(Accessed 1 December 2009)
42. Nowaczyk RH. Cognitive skills needed in computer programming, paper presented at the annual meeting of the southeastern psychological association. 1983;23-26.
 43. Leeper RR, Silver JL. Predicting success in a first programming course. 13th SIGCSE technical symposium on computer science education, Indianapolis, Indiana, United States. 1982;147-150. DOI:10.1145/800066.801357.
 44. Jones S, Burnett G. Spatial ability and learning to program. *An Interdisciplinary Journal on Humans in ICT Environments*. 2008;4(1):47-61.
 45. Özerbaş A. The Influence of computer assisted anchored instruction on student achievement, motivation and transfer skills. PhD dissertation. Ankara University, Ankara; 2003.
 46. Byrne P, Lyons G. The effect of student attributes on success in programming. 6th Annual conference on innovation and technology in computer education. Canterbury, United Kingdom. 2001;49-52. DOI:10.1145/377435.377467.
 47. Soloway E, Lochhead J, Clement J. Does computer programming enhance problem solving ability? Some positive evidence on algebra word problems. Seidel, R.J., Anderson, R.E., and Hunter, B. (Eds.), *Computer literacy: Issues and directions for 1985*. New York: Academic Press; 1982.


APPENDIX

Appendix A. An Example for Worksheet

Scratch Loop course study page: 2 Activity number: 4 subject: Loop

Purpose: 1. Students uses loops in a group repeating within the program
The cat character at the Scratch program will count with you the numbers from 1 to 20 by means of the algorithm at this activity. When the program finished, the screen image will be as in the Fig. 1.
To do this, it is necessary for you to drag required codes to the code domain and cause the algorithm to be functioned. Necessary steps for this job are indicated herein below.

Instruction

- 1) Use the button  in order to start the algorithm.
- 2) Create a variable, named as “number (sayı)” and give the value 1 to the variable.
- 3) Create a loop and ensure the cat character to count the numbers from 1 to 20.
- 4) Do not forget to increase the numbers by 1 within the loop.
- 3) The program will appear as follows when it is run. The character will count the numbers from 1 to 20 as is indicated in the following number

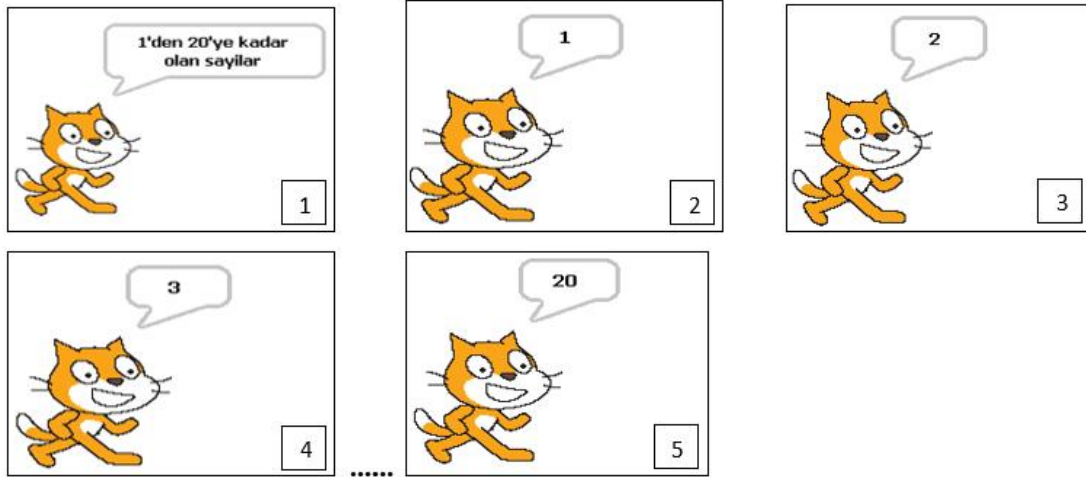


Fig. 1. Screen image for scratch activity

Appendix B. Pre-knowledge test

1. The steps that Ali may take while going from one place to other by bus are given herein below. Please insert numbers in blank places below in order to put the steps in correct order.

- Walk back when you approach the place where you will get off.
- Get off the bus when it stops.
- Wait at the bus stop for the arrival of the bus in the direction you want to go.
- Get on the bus.
- Walk to the bus stop.
- Press on the stop button.

2. Your printer is broken down while you want to get the printout of your homework. Pick up one of the problems that may be related to it and list a way out in steps.

3. Insert your information in five different fields due to define you in the following Table and write down the data types of variables in the domain.

Domain Name	Information	Data type
Your name:	...	Text
Your age:
Your date of birth (d/m/y):
Your place of residence:
Phone No:

4. Your computer will ask your exam point and give a value suitable for this point. If the point is less than 45, the message appearing on the screen will be "Failed", if over 45, however, "Passed". List the procedures due to be made by your computer in steps.

1. Step:
2. Step:
- ...

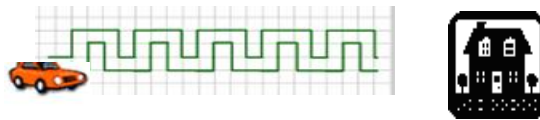
5. What will the result be when you apply the following procedures to x by giving the values from 1 to 8?

```
Repeat from x = 1 to 8
{
  If x is an odd number
    Multiply number x by 2
    print the result
  If x is an even number
    Multiply number x by 3
    print the result
}
```

6. If the following expressions are correct, mark them with (C), if false, however, with (F).

- a) If $x = ((125/5) + (3 + 4))$, $x = 5$ (..)
- b) For $x = 2$, $((x >= 2) \text{ and } (x <= 2))$ (..)

7. The route that should be followed by a driver, who got lost, to go home is given herein below. According to this, list up the steps that should be taken by the driver in the shortest possible way.



8. Write down the values i and j making the following expressions correct.

- a) If $i >= 1$ and $i <= 5$ $i = \dots\dots\dots$
- b) If $j >= 3$ and $j <= 7$ $j = \dots\dots\dots$

11. Does the following algorithm function correctly? Why?

```
Step 1: Start
Step 2: a = 1
Step 3: If (a > 3)
        {
Step 4:     Print "number "a" is
          smaller than 3."
Step 5:     Go to Step 8.
        }
Step 6: else
        {
Step 7:     Go to Step1
        }
Step 8: Stop
```

12. Let us say that the grade point average of Ayşe is "a" in English course. Draw an algorithm flow chart indicating "Your point is under 50" if the success point of Ayşe in English below 50 if the success point of Ayşe is more than 50 "Your point is over 50".

13. The forms of water in certain temperatures are given herein below. Let us say that the temperature of water is B. Create an algorithm by

the use of a conditional sentence indicating that if the value B;

- a. is less than 0, it will be "solid",
- b. between 0 and 100°, it will be "liquid",
- c. over 100°, however, then, "gas".

14. When the grades of students are entered by the use of a conditional sentence, write down a program algorithm indicating as follows;

- a. If the grade is = 1, "failed"
- b. If the grade is = 2, "pass"
- c. If the grade is = 3, "average"
- d. If the grade is = 4, "good"
- e. If the grade is = 5, "Very good"

15. Print the numbers ranging from 1 to 10 be written by the use of loop.

16. Let us say that the driving speed of a vehicle is a. Increase its driving speed 20 km/s until it exceeds 60 km/s. Stop the loop when the driving speed exceeds 60 km/s.

17. Draw the flow chart of an algorithm indicating the wording "I love my school very much" 20 times by the use of loop.

Appendix D. Post-Test (Computer-based Test)

1. List the steps that may be necessary for the multiplication of two variables referring to the values of 5 and 20 through the use of the program and create an algorithm.
2. Create two variables in number type and give value to such variables. Then add the numbers of these variables and display of the result on the screen.
3. Firstly, create a variable, named as number1 and assign a value of 20 to the variable. Afterwards, create a variable, named as number2 and assign a value of 30 to the variable. Subsequently, write on the screen the "multiplication of number1 and number 2 =". Ensure your algorithm to make the multiplication by the use of operators. Have the multiplication result be appeared on the screen.
4. Create a variable, named as number1 (sayı1) and assign a value of 15 to the variable before all else. Ensure that your algorithm checks the state of the number1 variable to be greater than 0 by the use of a condition sentence. If the number1 variable is bigger than 0, then have the wording "the number1 is a positive number", if not, "the number 1 is a negative number" be indicated on the screen.
5. Have the wording "Odd numbers from 1 to 10" be indicated on the screen. Cause the odd numbers from 1 to 10 be indicated on the screen by the use of loop.

© 2015 Gülmez and Özdenler; This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Peer-review history:
The peer review history for this paper can be accessed here:
<http://sciencedomain.org/review-history/10248>