Journal of Education and Learning; Vol. 6, No. 1; 2017
ISSN 1927-5250 E-ISSN 1927-5269
Published by Canadian Center of Science and Education

The Impact of Online Algorithm Visualization on ICT Students' Achievements in Introduction to Programming Course

Fatih Saltan[1]

[1] Department of Computer Education & Instructional Technology, Amasya University, Amasya, Turkey

Correspondence: Fatih Saltan, Department of Computer Education & Instructional Technology, Amasya University, Akbilek Mah. Muhsin, Yazıcıoğlu Cad. No: 7, 05100, Amasya, Turkey. E-mail: fsaltan@gmail.com

Received: October 18, 2016 Accepted: November 10, 2016 Online Published: November 21, 2016

doi:10.5539/jel.v6n1p184 URL: http://dx.doi.org/10.5539/jel.v6n1p184

Abstract

Online Algorithm Visualization (OAV) is one of the recent developments in the instructional technology field that aims to help students handle difficulties faced when they begin to learn programming. This study aims to investigate the effect of online algorithm visualization on students' achievement in the introduction to programming course. To achieve this goal, quantitative and qualitative investigations were conducted in a mixed method design. Participants of the study consisted of 40 ICT students who were taking the introduction to programming course for the first time in fall semester. Students were randomly assigned to treatment (n = 20) and control (n = 20) groups. During the first 4 weeks, the treatment group students participated in OAV. Concurrently, students in the control group were taught the semantics of programming and algorithm through traditional approaches. An achievement test consisting of six questions was used to measure ICT students' performance in computer programming at the end of the introduction to programming course. An open-ended survey and semi-structured interviews were also used to gain qualitative insight. The quantitative data were analyzed using t-test and ANOVA statistical analysis. The qualitative data were analyzed using content analysis techniques. Results showed that the experimental group, for which OAV treatment was implemented, had a higher mean score than the control group, for which traditional methods were implemented. There was a significant mean difference between the experimental group (M = 51.85, SD = 20.34) and the control group (M = 38.75, SD = 12.86). In qualitative analysis, five themes emerged. Students highlighted that OAV contributed to their algorithmic thinking (28%) and progressive thinking abilities (7%), and allowed for explorative learning (7%). Although their reasons varied, most of the students perceived OAV as an engaging instructional tool for learning computer programming.

Keywords: programming and programming languages, interactive learning environments, teaching/learning strategies

1. Introduction

Programming has become an important skill and strategic power in the 21st Century. We are surrounded by programmable devices, like smart phones, tablets, wearable technologies and home appliances, in our daily lives. However, learning programming and developing new programs is not easy, even with the most widely used programming languages (Gomes & Mendes, 2007; Tan, Ting, & Ling, 2009). Winslow (1996) highlighted that becoming an expert programmer takes 10 years of experience. Students who are just starting to learn programming face many difficulties. Ozmen and Altun (2014) specified these difficulties as programming knowledge, programming skills, understanding semantics of the program, and debugging. Before learning the concepts and syntax of a programming language, students must develop problem solving and algorithmic thinking skills (Tan, Ting, & Ling, 2009; Ersoy, Madran, & Gülbahar, 2006; McGill & Volet, 1997). In order to improve students' algorithmic skills, computer science educators have been working on Algorithm Visualization (AV) since 1980s (Brown, 1988). Algorithm visualization is an instructional approach which aims to enhance learning with technology that graphically represents how algorithms work (Hundhausen et al., 2002). One of the earliest examples of AVs was BALSA (Brown ALgorithm Simulator and Animator). BALSA allowed students to learn the concepts and fundamentals of computer science through system-generated visualizations (Brown & Sedgewick, 1984). Today there are hundreds of AVs that can be accessed freely, but only a few of them are suitable for educational purposes (Shaffer et al., 2010). In the 1990s, TANGO and Swan were developed to enable learners to create visual algorithm animations by utilizing C language (e.g., Stasko et al., 1998; Shaffer, Heath, & Yang, 1996). In these years, Pierson and Rodger (1998) created a Java-based system named JAWAA. They describe JAWAA as a "simple command language for creating animations of data structures and displaying them with a Web browser" (p. 267). In the present decade, more interactive and application-rich AV environments have been developed. For example, Alice was developed at Carnegie Mellon to enable beginners in computer programming to develop attractive three-dimensional graphic animations (Dann et al., 2000). Scratch is another recently developed AV which has gained great popularity. Scratch enables learners to develop "interactive, media-rich projects" (Maloney et al., 2010, p. 1). Like Alice and Scratch, Greenfoot was designed to teach programming in a visual way to those who have no prior programming experience (Kölling, 2010).

Generally, educators have used AVs to facilitate lectures, maintain individual learning, support laboratory assignments, and give homework (Hundhausen et al., 2002; Naps, 1990). In their meta-analysis study, Hundhausen and colleagues (2002) indicated that there are four main pedagogical approaches in algorithm visualization: Epistemic Fidelity, Dual-coding, Individual Differences, and Cognitive Constructivism. Epistemic Fidelity emphasizes encoding and transferring experts' mental model to learners through graphic illustration. This theory requires an excellent match between an expert's denotational semantics and the graphical representations. Dual-coding theory highlights two symbolic systems: verbal events and non-verbal events. The use of both words and graphics allows learners to create dual representations in their minds. The individual differences approach underlines the different needs, mental progress, abilities, and learning styles of each individual learner. The learning environment should address individual differences by providing different learning options. Cognitive Constructivism indicates that each individual constructs the knowledge in his/her mind. Each learner establishes the truth by themselves. In the learning process, students must be active and involved. In recent years, in addition to all of these approaches, the game-based learning approach has become a theoretical guide to AV programs. This approach emphasizes goal-oriented, active and enjoyable learning.

Since beginning in the mid-1970s, debates about effectiveness of AV has continued. Recently, developers have begun adapting algorithm visualization to online environments. Online algorithm visualization programs are accessible from computers and mobile devices via the internet. Without a doubt, OAV will be used more widely than the former programs. However, there is presently insufficient evidence about the effectiveness of these online learning tools. In this study, an online algorithm visualization program was employed and its impact was investigated in the introduction to programming course. The OAV used in this study was provided by a non-profit organization named "code.org". This organization was launched in 2013 and aims to teach computer programming and computer science through visualization in an online form. Code.org was chosen because it is online, free, and provides a user-friendly interface. The introduction to programming course was chosen because most students begin coding in such a course, which is a primary and mandatory course in many departments.

1.1 The Purpose of the Study

The aim of the study was to investigate the effect of an online algorithm visualization program on students' achievement in the introduction to programming course. Furthermore, this study compares the achievement of students exposed to OAV and students taught using the traditional approach. For the purpose of the study, the following specific research questions were addressed:

1) Is there a significant difference between the achievement of students in the control group and the experimental group in the Introduction to programming course?

2) Is there a significant difference between female and male students' achievement in the Introduction to programming course?

3) What do students perceive to be the contributions of OAV to their achievement in the Introduction to programming course?

2. Method

In order to explore the impact of an Online Algorithm Visualization program (OAV) on students' achievement in the introduction to programming course, a mixed method design was used (Tashakkori & Teddlie, 1998). Through collection of both quantitative and qualitative data, this method eliminates the weakness of one method with the strength of another (Creswell, 2013). On the quantitative side, an experimental research method with a control group was applied to investigate the effect of OAV applications on students' course achievement. A qualitative case study was performed to understand students' perceptions about the effect of OAV applications on their course performance and achievement. Quantitative and qualitative investigations were conducted concurrently with equal weights. The results of both types of data are presented in the results section and reviewed in the discussion section.

2.1 Setting

The study was conducted with ICT students in their second year at university. They were first enrolled in a mandatory programming language course that prepared them to begin coding and algorithm development. In the traditional design of the programming language course, students learn about algorithms and the semantics of computer programming by exploring several algorithm samples during first three or four weeks. Later, they begin coding with the C# programming language, one of the most powerful and widely used programming languages. In recent years, various instructional strategies and technological tools and applications have begun to be used to teach programming language. In this study, algorithm visualization program in an online environment was employed. This environment is provided by a non-profit organization named code.org. This organization was launched in 2013 and aims to teach computer programming and computer science through visualization. This environment teaches programming in a visual way those who have no prior programming experience by enable designing media-rich interactive projects. Code.org was selected for use in this study because it is online, free, and provides a user-friendly interface. This program provides various courses that can be chosen depending on the scope of the instructional objectives. In this study, the researcher utilized the "20 hour introduction course" on code.org. Moreover, this environment allows the instructor to observe his or her students' progress in real-time (see Figure 1).

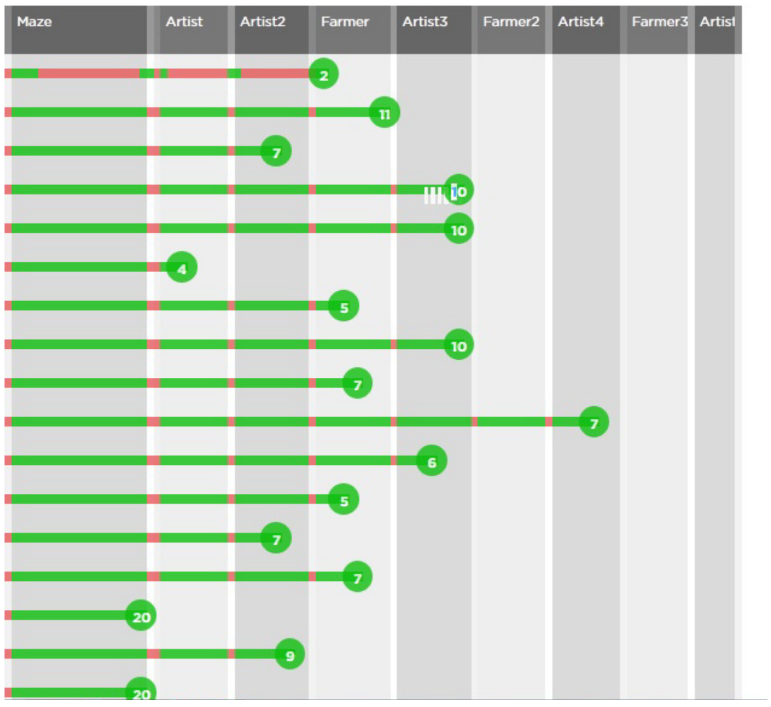

Figure 1. Students' progress

2.2 Samples

The participants of the study consisted of 40 ICT students who were taking the introduction to programming course for the first time in Fall semester. An experimental study with a post-test was conducted on the students, who were randomly assigned to treatment (n = 20) and control (n = 20) groups. In the treatment group there were 9 females and 11 males. The control group consisted of 6 females and 14 males.

2.3 Data Collection

A mixed-research design was used to address the research questions. The data were collected from multiple sources, including achievement test results (n = 40), open-ended survey responses (n = 20) and semi-structured interviews (n = 4). At the beginning of the course, during first 4 weeks, in order to apprehend the semantics of programming and algorithms, different instructional strategies were followed in the control and treatment groups.

During course hours, the treatment group students utilized the OAV application. All students in the treatment group individually completed the “20 hour introduction course” on OAV. During this process, the researcher tracked students’ progress and encouraged them. Concurrently, students in the control group were taught the semantics of programming and algorithms through traditional approaches. The instructor used the projector to provide algorithm samples and demonstrated special algorithm shapes on the blackboard. At the end of the 4-week period, only the students in the treatment group answered the open-ended survey. Interviews were also conducted with 4 volunteer students from the treatment group. Semi-structured questions were used and interviews were recorded via a sound recorder. Next, both control and treatment group students took a C# programming language course together during a six-week period. After the conclusion of the instruction, all students completed the VC# achievement test. The data collection process is represented in Figure 2.

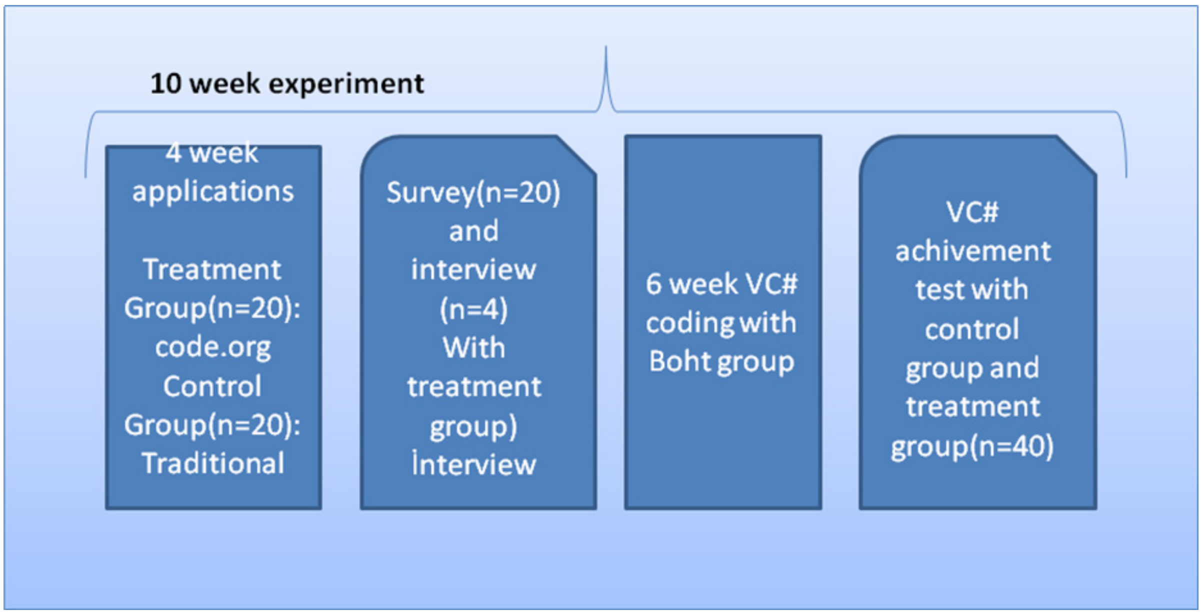
10 week experiment
4 week applications
Treatment Group(n=20): code.org
Control Group(n=20): Traditional
Survey(n=20) and interview (n=4) With treatment group) İnterview
6 week VC# coding with Boht group
VC# achivement test with control group and treatment group(n=40)

Figure 2. Data collection progress

2.4 Data Collection Instruments

The VC# achievement test, which consists of six questions, was used to measure ICT students’ performance in computer programming using C# programming language. This achievement test was adapted from the “Computer Programming Self-Efficacy Scale”, developed and validated (α = .98) by Ramalingam and Wiedenbeck (1998). This scale was originally a self-efficacy scale, not an achievement scale. In the present study, the items of the Computer Programming Self-Efficacy Scale were converted to six programming questions and validated for its relevance and concordance with the course syllabus by a commission at the university which consisted of an experienced programmer, an instructional technologist and a teacher educator. In the adapted test, students were not asked to rate their confidence, but rather to write related C# codes. A sample of the questions is provided in Table 1. The internal validity of the instrument was also evaluated by two teacher educators who have been teaching the same course in different universities. Both agreed that the achievement test was appropriate, and that the questions adequately covered the course content.

Table 1. Sample question

Original Item	Adapted Question in VC# achievement test
I could write a C++ program that displays a greeting message	Please write C# codes that displays a greeting message

2.5 Data Analysis

The quantitative data collected from the achievement test were analyzed using SPSS 20.0 for descriptive statistics and t-test analysis with 5% significance level. The analysis was performed to discover whether the difference in the scores of the achievement test on the control group that used traditional methods and treatment group that utilized the OAV reached a significant level. Additionally, ANOVA was conducted to investigate if there is a significant difference in terms of gender.

The qualitative data, collected from a questionnaire and interviews with students, were analyzed using content analysis techniques (Miles & Huberman, 1994). Content analysis is one of the most common qualitative data analysis methods which has been used in social science since the 1980s (Krippendorff, 2004). In this mixed method study, the qualitative aspect had equal importance because, as Patton (1990) asserted, the qualitative method allows researchers to gain in-depth information and understanding on the selected subjects. Answers to the open-ended questions and interview were transcribed and coded using Nvivo8 qualitative data analysis software. Appropriate themes were developed from related codes, which are internally consisted and distinct from each other (Marshall & Rossman, 2006).

3. Results

3.1 Research Question 1 and 2

The t-test results showed that the experimental group for which OAV treatment was implemented had a higher mean score than the control group for which traditional methods were implemented. There is a significant mean difference between the experimental group (*M* = 51.85, *SD* = 20.34) and the control group (*M* = 38.75, *SD* = 12.86) with the mean difference of 13.1 and large effect size; *t* (38) = 2.43, *p* > .05, *d* = .77. Table 2 represents *t*-test results.

Table 2. Result of achievement test

	t	df	Sig. (2-tailed)	Mean Difference	Effect Size (Cohen's *d*)
Achievement	2.43	38	.02	13.1	.77

An ANOVA test was also conducted to test if there is a significant mean difference between the experimental and control groups in terms of gender, to check whether gender has an influence on student achievement. Table 3 shows the results obtained as a result of the ANOVA test. These results indicate that there is no statistically significant difference for the overall model (F = 1.91, p > .05). There is a significant difference in terms of group (F = 4.86, p < .05), but there is no significant difference in terms of gender (F = .00, p > .05), and the interaction between group and gender is not statistically significant (F = .09, p > .05).

Table 3. Achivement results in terms of gender

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	1743,943[a]	3	581,314	1,907	,146
Intercept	75537,930	1	75537,930	247,745	,000
group	1482,604	1	1482,604	4,863	,034
gender	,017	1	,017	,000	,994
group * gender	27,734	1	27,734	,091	,765
Error	10976,457	36	304,902		
Total	94804,000	40			
Corrected Total	12720,400	39			

a. R Squared = ,137 (Adjusted R Squared = ,065).

3.2 Research Question 3

The qualitative analysis of the interviews and the responses of participants to open-ended questions revealed that all treatment group students perceived OAV to be a useful and effective tool for learning computer programming. Students mentioned various contributions of OAV. The resulting categories are listed in Table 4. There were seven themes which explain the contribution of OAV to students' learning of programming language (see Figure 3).

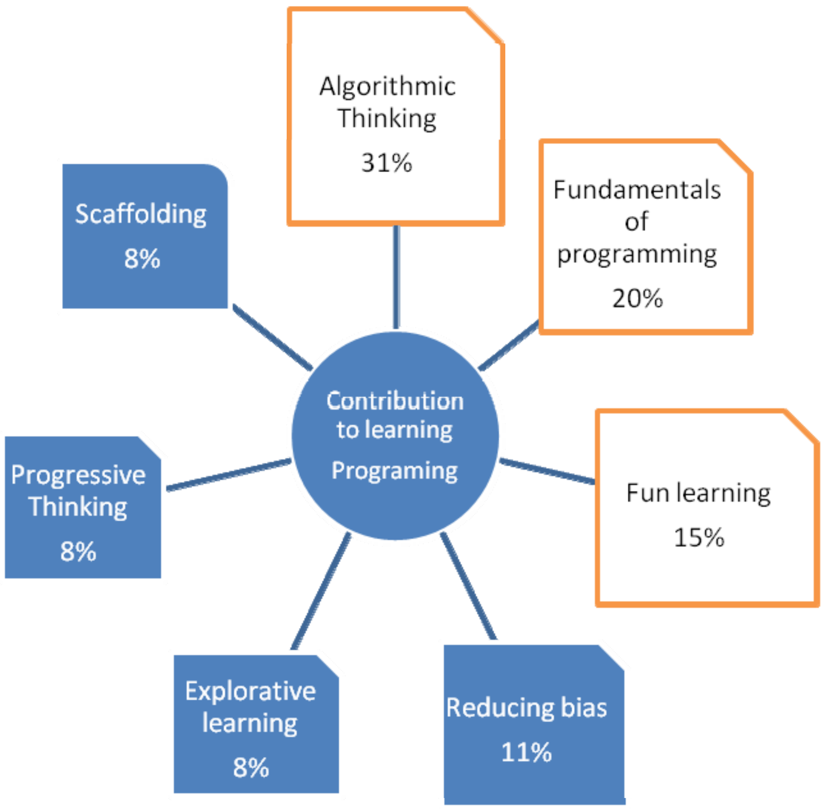

Figure 3. The themes which explain the contribution of OAV

Almost all students indicated that OAV enabled them to develop algorithmic thinking. They also stated that during the online activities, they began to understand the logic behind programs and began to think like a computer. They were able to complete the activities using this perspective. For example, one student said:

> "There is logic on the basis of programming. So we use our logic when writing a program. Code.org also has a parallel in relation to programming because everything made sense. Then it makes it easy for a person to begin to write programs using Code.org."

Students also indicated that OAV enabled them to understand the fundamentals of programming. All of the four interview participants highlighted that OAV activities prepared them for programming and they believed that this was an excellent opportunity which must provided all students who are beginning to learn a programming language. One of them stated:

"The Code.org application was very useful. I think I started learning programming from the most basic. I do not have any previous programming experience. OVA enable me understand coding. I gained benefits for myself."

About half of the students clearly stated that using OAV was enjoyable. They highlighted that developing a program without coding was wonderful and provided them an enjoyable learning experience. One student indicated, "I think that makes learning more enjoyable. I think we are more willing. Instead of writing code to begin with, it was better to first do more basic applications on code.org". Several students also reported that at the beginning of the course they had a strong bias against coding. Analysis indicated that OAV activities did not reduce this bias completely, but it fostered positive feelings towards programming. For example, one student said, “I had no previous programming experience. This application somewhat reduced my bias against coding. I developed a more moderate approach to the course which I initially did not like”. Several students, including two of the interview participants, requested two main things. First, beyond the current OAV, they want a comprehensive application which enables them to learn not only the algorithm and logic behind programming, but also coding and programming language itself. Second, they wish to encounter OAV in primary school. One of them stated, “If programming courses were given in primary school, students would not be afraid of coding and would grow up as people who like programming and computers”. Lastly, the other three important themes which emerged in the data analysis were explorative learning, progressive thinking and scaffolding. Students indicated that they had the opportunity to learning through discovery, which made them more active learners. Participants also said that during the activities, they learned to put visual commands in the right place and in the right order, otherwise the applications would not work. These OAV activities improved their progressive thinking skills. Students also mentioned the efficacy of scaffolding on OAV, including videos and clues. They indicated that those scaffolding elements were very helpful and instructive. In this regard, one student said, “Code org. is a good practice environment that enables users to learn coding following a unique strategy that involves tips and videos”.

Table 4. Perception of students regarding effectiveness of OAV

Themes/Categories	Frequency (n = 152)
Algorithmic Thinking	20 (28%)
Fundamentals of programming	13 (18%)
Fun learning	10 (14%)
Reducing bias	7 (10%)
Requests	6 (8%)
Explorative learning	5 (7%)
Progressive Thinking	5 (7%)
Scaffolding	5 (7%)
Total	71 (100%)

4. Discussion and Conclusion

This study aimed to investigate the effect of an online algorithm visualization program on students’ achievement in the introduction to programming course. The achievement of students using OAV was compared with the achievement of students learning through the traditional approach. In order to achieve this goal, a mixed method research design was followed. Results showed that the experimental group, for which OAV treatment was implemented, had a higher mean score than the control group, for which traditional methods were implemented. A significant mean difference between the groups was found. The literature about computer programming is extensive in both the field of education and the field of psychology (Robins, Rountree, & Rountree, 2003). While some previous studies reveal evidence about the effectiveness of AVs, many do not. For example, Lewis (2010) examined and compared the impact of two of the recently developed AV applications and found out that AVs makes programming easier, improve learning outcomes to some extent, and support students’ confidence. Cooper, Dann and Pausch (2003) used Alice, one of the most popular AVs, in the CS1 course. They highlighted that compared to other students, the achievement of the participants who used Alice was significantly higher. In the present study, it was also investigated whether gender has an influence on student achievement. Results showed that there is no significant difference in terms of gender. Some studies on computer programming point out the different achievement levels of males and females (Beyer et al., 2003; Lips & Temple, 1990). In one of the earliest studies, Carter and Jenkins (1999) found a relation between learning approach and gender regarding achievement in computer programming. However, in this study, a meaningful effect of gender was not found. It is important to clearly indicate that qualities of programming courses effects achievement of students overall both male and female (Linn & Dalbey, 1989).

Qualitative analysis showed that all treatment group students perceive OAV to be a useful and effective tool for learning computer programming. Students indicated various contributions of OAV. These contributions of OAV to students' learning of programming language fell under seven themes: algorithmic thinking, fundamentals of programming, fun learning, reducing bias, explorative learning, progressive thinking, and scaffolding. These themes also demonstrate why students perceive OAV to be effective regarding learning computer programming. Overall, these results corresponded with the results of Genç and Karakuş' study (2011). This study showed that most students (79%) who experience AV perceive it to be a fun learning application. Participants also thought that AV is easy to use for learning the fundamentals of programming.

Overall, this study showed that online algorithm visualization is highly promising. OAV provided students explorative learning and progressive thinking opportunities that are hard to gain through traditional approaches. Video tutorials provided effective scaffolding, while online accessibility ensured mobility and flexibility. This surely affects students' confidence positively and significantly contributes to participants' course achievement.

References

Beyer, S., Rynes, K., Perrault, J., Hay, K., & Haller, S. (2003). Gender differences in computer science students. *ACM SIGCSE Bulletin*, *35*(1), 49-53. https://dx.doi.org/10.1145/792548.611930

Brown, M. H. (1988). *Algorithm Animation*. MIT Press.

Brown, M. H., & Sedgewick, R. (1984, January). A system for algorithm animation. In *SIGGRAPH'84 Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 177-186). ACM. https://dx.doi.org/10.1145/800031.808596

Carter, J., & Jenkins, T. (1999). Gender and programming: What's going on? *ACM SIGCSE Bulletin*, *31*(3), 1-4. https://dx.doi.org/10.1145/384267.305824

Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. *ACM SIGCSE Bulletin*, *35*(1), 191-195. https://dx.doi.org/10.1145/792548.611966

Creswell, J. W. (2013). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Ersoy, H., Madran, R. O., & Gülbahar, Y. (2006). Programlama Dilleri Öğretimine Bir Model Önerisi: Robot Programlama. In *Akademik Bilişim'07 Konferansı*. Kütahya.

Genç, Z., & Karakuş, S. (2012). TASARIMLA ÖĞRENME: EĞİTSEL BİLGİSAYAR OYUNLARI TASARIMINDA SCRATCH KULLANIMI. In *5th International Computer & Instructional Technologies Symposium*.

Gomes, A., & Mendes, A. J. (2007, September). Learning to program-difficulties and solutions. In *International Conference on Engineering Education*. Coimbra, Portugal.

Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, *13*(3), 259-290. https://dx.doi.org/10.1006/jvlc.2002.0237

Kölling, M. (2010). The greenfoot programming environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 14. https://dx.doi.org/10.1145/1868358.1868361

Krippendorff, K. (2004). *Content Analysis: An Introduction to Its Methodology* (2nd ed.). Thousand Oaks, CA: Sage.

Lewis, C. M. (2010, March). How programming environment shapes perception, learning and goals: Logo vs. scratch. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 346-350). ACM. https://dx.doi.org/10.1145/1734263.1734383

Linn, M. C., & Dalbey, J. (1989). Cognitive consequences of programming instruction. In E. Soloway, & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 57-81).

Lips, H. M., & Temple, L. (1990). Majoring in computer science: Causal models for women and men. *Research in Higher Education*, *31*(1), 99-113. https://dx.doi.org/10.1007/BF00992559

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 16. https://dx.doi.org/10.1145/1868358.1868363

Marshall, C., & Rossman, G. B. (2006). *Designing Qualitative Research* (4th ed.). Thousand Oaks: Sage.

McGill, T. J., & Volet, S. E. (1997). A Conceptiual Framework for Analyzing Students' Knowledge of Programming. *Journal of Research on Computing in Education*, *29*(3), 276-297. https://dx.doi.org/10.1080/08886504.1997.10782199

Miles, M. B., & Huberman, A. M. (1994). *Qualitative Data Analysis: An Expanded Sourcebook* (2nd ed.). California: Sage.

Naps, T. L. (1990). Algorithm visualization in computer science laboratories. *ACM SIGCSE Bulletin*, *22*(1), 105-110. https://dx.doi.org/10.1145/319059.323422

Ozmen, B., & Altun, A. (2014). Undergraduate Students' Experiences in Programming: Difficulties and Obstacles. *Turkish Online Journal of Qualitative Inquiry*, *5*(3), 9-27. https://dx.doi.org/10.17569/tojqi.20328

Patton, M. Q. (1990). *Qualitative evaluation and research methods* (2nd ed.). Thousand Oaks, CA: Sage.

Pierson, W. C., & Rodger, S. H. (1998, March). Web-based animation of data structures using JAWAA. In *SIGCSE 98 Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education* (pp. 267-271). ACM. https://dx.doi.org/10.1145/273133.274310

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), 367-381. https://dx.doi.org/10.2190/C670-Y3C8-LTJ1-CT3P

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137-172. https://dx.doi.org/10.1076/csed.13.2.137.14200

Shaffer, C. A., Cooper, M. L., Alon, A. J. D., Akbar, M., Stewart, M., Ponce, S., & Edwards, S. H. (2010). Algorithm visualization: The state of the field. *ACM Transactions on Computing Education (TOCE)*, *10*(3), 9. https://dx.doi.org/10.1145/1821996.1821997

Shaffer, C. A., Heath, L. S., & Yang, J. (1996). Using the Swan data structure visualization system for computer science education. In *ACM SIGCSE Bulletin* (Vol. 28, No. 1, pp. 140-144). ACM. https://dx.doi.org/10.1145/236462.236527

Stasko, J., Domingue, J., Brown, M. H., & Price, B. A. (Eds.). (1998). *Software visualization: Programming as a multimedia experience*. MIT press.

Tan, P. H., Ting, C. Y., & Ling, S. W. (2009, November). Learning difficulties in programming courses: Undergraduates' perspective and perception. In *Computer Technology and Development, 2009. ICCTD'09. International Conference on* (Vol. 1, pp. 42-46). IEEE.

Tashakkori, A., & Teddlie, C. (1998). Mixed Methodology: Combining Qualitative and Quantitative Approaches. In *Applied Social Research Methods Series*. Sage, Thousand Oaks, CA.

Winslow, L. E. (1996). Programming pedagogy—A psychological overview. *ACM SIGCSE Bulletin*, *28*(3), 17-22. https://dx.doi.org/10.1145/234867.234872