Informatics in Education, 2020, Vol. 19, No. 4, 589–609

DOI: 10.15388/infedu.2020.26

Teaching Loops Concept through Visualization Construction

Ibrahim CETIN
Facuty of Education, Bolu Abant Izzet Baysal University, Turkey
e-mail: ibretin@hotmail.com

Received: February 2020

Abstract. Loops concept is one of the basic programming concepts. Students have difficulties in learning loops concept. Helping learners understand loops is an important task. Visualization is one of the ways to help students improve their understanding. The aim of the study was to construct and evaluate a visualization based instruction related to loops. A mixed method study was conducted. In the experimental phase of the study, the effect of visualization based instruction on pre-service teachers' achievement, perceived learning and programming attitude was examined. In the qualitative phase of the study, the main purpose was to get more in depth data related to experimental phase. Visualization based instruction helped pre-service teachers improve their understanding of loops concept.

Keywords: Loops concept, visualization, programming pedagogy, mixed method design, dual effect.

Introduction

Cuny, Snider, and Wing (2010) considered computational thinking as a problem solving process and defined it as "the thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (as cited in Wing, 2011, p. 20). Depending on their research experience in Scratch, Brennan and Resnick (2012) considered computational thinking in terms of programming constructs and their use in problem solving in a social environment. They provided three facets of computational thinking as:

(i) Computational concepts (programming notions).
(ii) Computational practices (practices of problem solving).
(iii) Computational perspectives (reflection on computing practices).

Programming is one of the best ways to let students experience power of computational thinking (Barr and Stephenson, 2011). However, programming can pose cognitive

problems for students. There is abundant literature contending that students have difficulties in learning programming (Duboulay, 1986; Robins, Rountree and Rountree 2003; Wiedenbeck, Ramalingam, Sarasamma, and Corritore, 1999).

Basic constructs of programming constitute cognitive challenge for students, for example, students have difficulties in learning loops (Ginat, 2004; Putnam, Sleeman, Baxter and Kuspa 1986; Seidman, 1989). Kordaki, Miatidis and Kapsampelis (2008) argued that students were not able to value the role of nested loops in the context of Bubble sort. Teague and Lister (2014) gave university students, who took 12 weeks, 20 weeks or 24 weeks of introductory programming instruction, the code that takes an array of numbers including five integers and move all of the elements of the array one place to the right and replace the first element with the last one. Then they asked students to write a code that reverses the effect of the given code, i.e. cancel the effect of the code by reversing to get the original state of the list. Only a small percentage of students were able to correctly handle the problem. Izu, Weerasinghe and Pope (2016) echoed the same result by stating that only a small percentage of students were able to correctly handle the reversibility problem. Moreover, there are studies that show students have misconceptions related to loops. Cetin (2015) showed that some of the mechanical engineering students who are taking introductory programming course perceived a nested loop as if two loops are running simultaneously. Similarly, Izu, Weerasinghe, and Pope (2016) reported the same misconception for university students. Interestingly, the same misconception was reported for middle school students (Mladenovic, Boljat and Zanko, 2018). Moreover, Du Boulay (1986) contended that novices were not able to see that control variable is increased in each step of for loop and they believed that while loop terminates when the loop condition is changed. Ginat (2004) showed that students have problems in constructing associations between range conceptions and loop boundaries.

Cetin (2015) described Pre-action, Action, Process and Object stages for loops concept. These stages correspondingly starts from relatively concrete and goes to relatively abstract. The details of each stage are given as follows:

(i) At the Pre-action stage, an individual is not able to write a loop code in a syntactically correct way. Although the individual is aware of the fact that the given task requires iteration, he/she is not able to use loops to develop running codes.

(ii) At the Action stage, the individual explicitly performs each line of instruction inside the loop. She/he can initialize control variables, check test conditions, repeat the body of the loop while test condition is true (but she/he still explicitly performs each line of the instruction in the body) and update the control variables. The individual can correctly write a loop code, but he/she can only solve easy problems by using iteration at this stage.

(iii) At the Process stage, action of the previous stage is interiorized. Conception of loops has dynamic flavor at this stage. The individual can perform action of the previous stage, but this time she/he does not need to perform them explicitly. Rather she/he can imagine hem being performed. She or he can perform the instructions after the loop (called afterward) but needed for the completion of the task, e.g. printing the result found in the body. The individual can solve relatively complex problems by using iteration.

(iv) At the Object stage, process described in the previous stage can be seen as a completed totality having input (boundaries of the control variable are determined), process (the loop is executed while test condition is true) and output (afterward is executed). Rather than a code having individual lines, loops (one-level) can be considered as a single line of command having some purpose and can be put in the body of another loop to get a nested (two-level) loop. A two-level loop can be put into the body of a one-level loop to get a three-level loop. This can be proceeded to get an n-level loop.

In order to bring the work on learning difficulties related to loops concept to the stage at which it is useful in practice, several questions should be considered such as: "how to help students to develop their understanding", "how to help them to overcome their misconceptions" or "how to help students so that they do not construct misconceptions". Despite the importance of loops as a computational concept, our knowledge base about how to facilitate students' understanding is limited. Visualization is one of the suggested tools that can be used to help learners overcome their cognitive difficulties in learning programming concepts (Ben-Ari *et al.*, 2011; Sorva, Lönnberg and Malmi, 2013; Stasko, Kehoe and Taylor, 2001; Velazquez-Iturbide and Perez-Carrasco, 2016).

Price, Baecker and Small (1998, p.4) focused on the visualization production and its purpose and defined the software visualization (SV) as "the use of crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software." They classified algorithm visualization (AV), dealing with visualization of higher level abstractions, and program visualization (PV), dealing with the actual programs, as two subfields of SV. Unlike Price, Baecker and Small's (1998) product based definition Zazkis, Dubinsky and Dautermann (1996, p. 441) had a cognitive point of view and defined visualization as "an act in which an individual establishes a strong connection between an internal construct and something to which access is gained through the senses." The definition sees visualization as an act not the product and it is a twofold definition. It both includes the construction of mental structures with the help of 'external' objects or events (e.g. a picture or an animation) and the construction of 'external' objects or events with the help of mental structures. However, the definition excludes the construction of mental images from pure mental images and transfer of an external image from one place to other (e.g. from paper to computer screen) without applying for mental constructions.

Hundhausen, Douglas and Stasko (2002) contended that studies imposed students to view visualizations did not report significant learning gains. In contrast, the studies in which students actively involved in creating the visualizations reported the highest percentage of significant results. Considering that, Naps *et al.* (2003) suggested an engagement taxonomy ranging from no AV to presenting AV that is constructed by the students:

(1) No viewing,
(2) Viewing,

(3) Responding,
(4) Changing,
(5) Constructing.
(6) Presenting.

Highlighting active engagement is important but it does not guarantee qualified active engagement experience. Zazkis, Dubinsky and Dautermann (1996) went one step further and studied on ways to help students improve their understanding while they are actively involved with visualization. They proposed a heuristic model, called V-A (Visual – Analytic), that integrates visual and analytical reasoning in constructing rich understanding of concepts. The model name is originally abbreviated as VA, but V-A was used in this study to better differentiate it from AV (algorithm visualization). According to V-A model, learning starts with an act of visualization. This is followed by an analysis in which an individual reason about the visualization and construct some internal structures related to concept under investigation. Then, the second step of visualization that represents richer understanding of the concept since individual improved her understanding in the first step of analytic thinking. After that the individual reasons about the new visualization to improve her understanding that is called the second step of analysis. In the process, each act of analysis helps the individual to develop new richer visualization that results in a more sophisticated analysis. This process ends with the integration of visual and analytical understanding into one coherent and (relatively) abstract structure.

Scratch was used as a SV construction tool in this study (Merino, Ghafari, Anslow and Nierstrasz, 2018; Sorva, Karavirta and Malmi, 2013). Scratch can be described as low floor and high ceiling block based coding environment (Resnick *et al.*, 2009). Students used Scratch to construct SVs. V-A model of Zazkis, Dubinsky and Dautermann (1996) was used for this purpose. V-A model proposes stages in which students improves and integrates the visualization of a concept and the analytical understanding of the concept. The Scratch and V-A model provides a unique way for programming instruction. This is the first study in which the aforementioned instruction was constructed and the effect of it was explored. The details of the instruction is given the intervention subsection of the study. Therefore, the purpose of the study is to examine the effect of Scratch-based SV construction guided by the V-A model on pre-service teachers' loops concept achievement, perceived learning, and their attitudes toward programming.

Materials and Methods

Mixed methods research utilizes qualitative and quantitative research approaches in combination. This combinatorial approach may provide unique insights and understandings that might be difficult to develop with a single approach (Johnson and Onwuegbuzie, 2004). This study utilized two-phased embedded experimental model (Creswell and Clark, 2007). Qualitative data has a supportive role in the embedded experimental

model. An experimental design with pre-test/post-test control groups was used in the first phase of the study. The aim of the first stage was to examine the effects of Scratch-based and V-A guided SV construction on students' loops concept achievement, perceived learning and programming attitude. The second stage used qualitative approach. The aim of the second stage was to explore the results of the first stage of the study. The research questions guided the study are (i) What is the effect of Scratch-based SV construction guided by the V-A model on (a) pre-service teachers' loops concept achievement, (b) perceived learning, and (c) their attitudes toward programming and (ii) How do qualitative results explain the experimental outcomes? The design of the study is shown in Table 1.

At the outset, pre-service teachers were randomly assigned to experimental and control groups. The intervention, in which students were given loops concept, was conducted over three weeks. The instruction consisted of three main components: classroom sessions, laboratory sessions and weekly homework. The only difference between the control and experimental groups lies in their laboratory sessions. The instructional cycle began with a 3-hour classroom meeting. The teaching approach and the instructor was the same in both groups. The instructor mainly used lecturing, discussion and problem solving in classroom meetings. After the classroom meetings students attended two-hour computer laboratory. Pre-service teachers studied in groups in the computer laboratory. Experimental group pre-service teachers constructed SVs of programs written in C by using Scratch in the computer laboratory. Visualization construction was guided by V-A model (Zazkis, Dubinsky and Dautermann, 1996). Control group pre-service teachers solved assigned problems by using C. The weekly instructional cycle continued with homework including relatively classical programming problems. The homework was the same for both groups. Pre-service teachers individually submitted homework within one week. After three weeks of instruction pre-service teachers were given an achievement test, a practice test, a perceived learning scale and an attitude scale as posttests in both groups. After the quantitative phase of the study, 9 pre-service teachers from both groups were randomly selected for the qualitative phase. Semi-structured interviews, each of which lasted about 23 minutes, were conducted and audiotaped.

Table 1
The design of the study

	Treatment	Posttests	Interview
Experimental group	V-A guided visualization	Achievement test Practice test Perceived learning Attitude scale	Semi-structured interview
Control group	Traditional instruction	Achievement test Practice test Perceived learning Attitude scale	Semi-structured interview

Subject

The participants of this study consisted of 53 pre-service teachers, 29 male and 24 female. They attended a four year program including both education and computer science related courses. They were studying to obtain a certification in computing education from a public university. They are expected to teach computing related courses after the graduation. They enrolled in an introductory programming course that was offered in the first semester of the second year. This is the first course on programming. The content of the course require students to solve small scale problems in Scratch and C. Four weeks of the semester were dedicated to Scratch instruction after which pre-service teachers develop an educational game or simulation. The remaining 10 weeks of the semester were dedicated to C instruction in which students learn the basic constructs of the programming language to solve small scale problems. When the study began students had already completed the Scratch sessions and learned variables, conditional statements and arrays in C.

Intervention

Instruction for both groups involved classroom phase, laboratory phase and homework. The same instructor instructed both experimental group including 27 pre-service teachers and control group including 26 pre-service teachers for three weeks in this study. The instruction included for, while and do-while loops. Pre-service teachers in both groups attended three-hour classroom phase in both group. The same type of instruction was provided for both of the groups in the classroom phase. Classroom phase started with a question with the purpose of reviewing pre-service teachers' previous knowledge and forming the base for the concept of the week. After questioning, instructor explained the main ideas and key concepts related the topic of the week in front of the board. This was followed by a demonstration of a simple code related to loops with the aim of teaching the syntax of the programming construct in consideration. Then instructor provided a programming problem related the topic of the week and gave pre-service teachers a few minutes to think on the problem. Instructor solved the problem on the computer whose screen is projected onto the board. After live coding, pre-service teachers were provided a programming problem. The problem was considered by the whole class. Pre-service teachers were given programming problems at the end of the classroom phase. Pre-service teachers studied in groups, including two or three members, to solve the problems. The instructor showed how to solve some of the problems in front of the board when he felt necessary.

Following the classroom phase, pre-service teachers in control group attended two-hour laboratory phase each week in which the instructor was present. The beginning of the laboratory phase was dedicated to summarization of the topics considered in the classroom meeting. Then pre-service teachers were given lab-sheets consisted of programming problems on the topic of the week. They studied in their groups to solve the

problems. Each group had one computer. Study groups were encouraged to negotiate their ideas in the computer laboratory. The instructor was present to help students in their learning in the computer laboratory. However, instructor tried not to provide whole solution to the pre-service teachers. He showed some hints and general approaches to solve the problem. The instructor monitored the group and individual work during the laboratory phase. He gave group and individual feedback. The instructor tried to enrich the discussions in the groups. Finally, pre-service teachers were given a homework including relatively classical questions. They were required to submit their homework within one week.

After the classroom meeting, experimental group pre-service teachers attended two-hour laboratory phase each week. The same instructor was present during this phase. Pre-service teachers constructed a visualization related to topic of the week in each computer laboratory. They used Scratch to construct their visualization in the laboratory. Since pre-service teachers had already learned Scratch as a part of regular curriculum before the intervention, they were not given additional Scratch instruction. Visualization activities were developed considering the V-A model of Zazkis, Dubinsky and Dautermann (1996). The model supports the cyclic mode of visual thinking followed by analytical thinking. At each new cycle individuals have a richer version of the concept in their mind, i.e. there is a progression from (relatively) concrete to (relatively) abstract.

In the first week of the laboratory sessions, pre-service teachers were given the following two codes and they were asked to construct their visualizations. The content of the array in code 2 was given by the instructor to each group and each group had a different integer array.

Code 1:

```
while (i<20){
    if (i%2 == 0)
        printf("%d", i);
    i++;
}
```

Code 2:

```
for (i=0; i<6; i++){
    if (a[i] >= a[i+1]){
        ara = a[i];
        a[i] = a[i+1];
        a[i+1] = ara;
    }}
```

The visualization construction had three steps: (i) create a storyboard of the visualization and discuss it with the instructor, (ii) develop the visualization, (iii) get feedback from the instructor and make changes on the visualization if necessary.

In the second week, pre-service teachers were given the following code and they were asked to construct its SV. The steps of the SV construction were the same for the activity and pre-service teachers used the same strategies as they did in week 1 to construct their SVs.

```
for(j=0; j<4; j++){
    for(i=0; i<4-j; i++){
        if (a[i] >= a[i+1]){
        ara = a[i];
        a[i] = a[i+1];
        a[i+1] = ara;
        }
    }
}
```

The visualization activity of week 2 requires to think on a nested loop while visualization activity of week 1 requires to think on a single loop. In constructing SV of week 2 pre-service teachers need to reflect on the looping of inner and outer loop. This can help them to develop their understanding of loops concept.

In the third week, pre-service teachers were given the Bubble sort code of the previous week and required to construct visualization of it. However, they were not allowed to have the approach in which states of storyboard are transferred to the Scratch environment as they did last two weeks; they were not allowed to only use goto, wait and glide blocks. They had to consider loops from more abstract perspective to handle the task. Their Scratch code should automatically animate the Bubble sort. For this purpose, (i) they needed to use loops, the list and variables of Scratch to sort the given array in Scratch, i.e. they needed to implement Bubble sort in Scratch and (ii) they needed to coordinate Bubble sort script of Scratch with the visualization of the C code. Visualization construction had mainly three steps:

(i) Plan how to develop the visualization and discuss it with the instructor.
(ii) Develop the visualization.
(iii) Get feedback from the instructor and make changes on the visualization if necessary.

Fig. 1 is an example screenshot of stage of Scratch.

The stage is composed of two parts. The left part displays the current state in the animation of the C code. The right part displays the current state of variables i, j, ara, and the array named a (list in Scratch terms) that will be sorted at the end of the visualization. The right side itself belongs to the implementation of Bubble sort in Scratch, i.e. they are real Scratch list and variables and list a is sorted with a real Scratch Bubble sort script. Each line of code on the left side is a sprite in Scratch. To develop the left side pre-service teachers needed to coordinate Bubble sort with the highlighting in the animation. The visualization activity of week 3 is different from visualization activities of week 2 and week 1. Pre-service teachers needed to reconstruct Bubble sort in Scratch and coordinate the left and the right side of the stage.

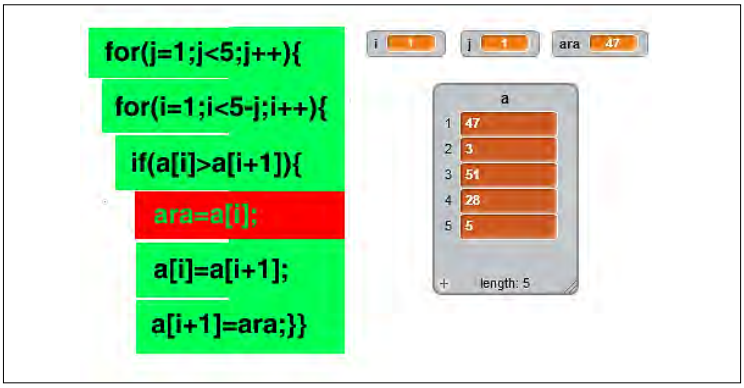

Fig. 1. Screenshot example for week 3.

Instruments

The study included five different tools to gather data that were composed of (i) a computer programming attitude scale (CPAS), (ii) a practice test (PT), (iii) an achievement test (AT), (iv) a perceived learning scale (PLS), and (v) a semi-structured interview protocol. The AT including 20 multiple choice questions was constructed by the researcher of the study considering the objectives of the course. The AT was used as a posttest to assess pre-service teachers' understanding of loops concept. Each item in the AT had one correct and four incorrect answers. Each correct item was given 5 points and the minimum and maximum points that a pre-service teacher can get are 0 and 100 respectively. The questions in the questionnaire were checked by a language expert in terms of comprehensibility and grammatical aspects. Two domain experts considered the content validity of the AT. They contended that questions in the achievement test is appropriate to the course content and pre-service teachers' grade level. The AT was conducted to 212 university students who already took at least one introductory programming course before the study has begun. The reliability was found to be .83. The following is a question from the achievement test.

Q. Which of the following choices correctly represents the computation stored in the variable *result* after the following code executed?

int i, inter, n, result = 0;
scanf("%d", &n);
for (i = 2; i <= n; i++){
  inter =i ∗ (i - 1);
  result = result + inter;
}
	a) 1+3+5+…+(2*n-1)
	b) 1!+2!+3!+…+n!
	c) 1∗2∗3∗…∗(n-1)∗n
	d) 1∗2+2∗3+3∗4+…+(n-1)∗n
	e) 1∗3+5∗7+…+(2∗n-3)∗(2∗n-1)

The PT included four open ended questions. The purpose of the PT was to assess teacher candidates' code writing practice related to loops concept. Two independent researchers established the content validity of practice test by considering it in terms content and pre-service teachers' grade level. A language expert considered the questions in the practice test in terms of their comprehensibility and grammatical aspects. The practice test was given in the computer laboratory. Pre-service teachers were allowed to execute their codes. They gave their source codes to the instructor at the end of the test. An expert and the researcher of the study scored practice tests and .89 (Pearson's r) interrater agreement was achieved. The following question is an from the PT.

Q. A number is called Armstrong number if it is equal to the sum of its own digits each raised to the power of the number of digits, e.g. 371 is an Armstrong number since $3^3+7^3+1^3=371$. Please write a C code that finds if a given number is Armstrong or not.

The PLS was constructed by Rovai, Wighting, Baker, and Grooms (2009). The scale was developed to measure students' perceived learning in a course. It was contended that adult learners themselves can judge their learning in a course (Rovai, Wighting, Baker, and Grooms, 2009). So, their learning can be assessed depending on self-reports of the learners. The scale consists of nine 5-point Likert-type items. The total score that a teacher candidate can get from the PLS ranges from 9 to 45. The reliability of the PLS was originally found to be .79. The scale was then adapted into Turkish by Top, Yukselturk and Inan (2010). The reliability of translated survey was determined to be .81.

The CPAS was constructed by Cetin and Ozden (2015). It was constructed to assess university students' attitudes towards programming. It includes 18 5-point Likert-type items. The scale has three dimensions: cognition, behavior and affection. The total score can a pre-service teacher get from the scale ranges from 18 to 90. Cronbach alpha coefficients were originally determined to be .80 for the cognition, .90 for the behavior, .90 for the affection, and .93 for the scale.

Nine participants from each group (18 in total) were interviewed after the intervention. A semi-structured interview protocol was utilized to collect data. The aim of the interviews was to collect in-depth data to reflect on the results of the intervention. Teacher candidates were given programming questions and asked to solve them in the interview. The interviews were audio recorded and transcribed. The data were analyzed with the guidance of qualitative content analysis. Qualitative content analysis is "… a method for systematically describing the meaning of qualitative material" (Schreier, 2012, p.1). Both inductive and deductive approaches were used for category construction. A sample of data were coded by an independent researcher and the researcher of this study. Then they came together to consider discrepancies related to coding and coding scheme. They repeated cycle of coding, discussing and solving issues until they had reached an agreement. Then, 10% of the data were independently coded by two researchers to assess the inter-coder reliability. The agreement ratio was determined to be .88. Researchers continued to discuss and resolve the discrepancies until unanimous agreement was achieved. Finally the researcher of this study coded all of the data.

Results

Quantitative Results

The quantitative results examine the effect of intervention on pre-service teachers' AT, PT, PLS and CPAS scores. Table 2 summarizes the experimental and pre-service teachers' AT, PT, PLS and CPAS scores.

An independent sample t-test analysis was conducted to examine whether there was a significant mean difference between the groups in terms of pre-service teachers' AT scores after the treatment. It was found that mean difference between the experimental (M = 53.2, SD = 14.4) and the control group (M = 42.5, SD = 15.1) in terms of pre-service teachers' AT scores at the end of the instruction was significant. The experimental group performed significantly better than the control group; $t(51) = -2.630$, $p = .011$.

An independent sample t-test was utilized to compare mean PT scores of the control and the experimental group after the treatment. It was found that the mean difference between the experimental (M = 50.5, SD = 16.7) and the control group (M = 38.0, SD = 16.2) in terms of their PT scores at the end of the instruction was significant ($t(51) = -2.759$, $p = .008$).

An independent sample t-test analysis was conducted to compare mean perceived learning scores of experimental and control group students. It was found that mean difference between the experimental (M = 34.6, SD = 5.3) and the control group (M = 31.7, SD = 5.6) in terms of students' PLS scores at the end of the instruction was not significant ($t(51) = -1.881$, $p = 0.66$).

The effect of instruction on students' attitudes toward computer programming was examined by one-way MANOVA. Before carrying out main analysis, its assumptions were tested. Skewness and kurtosis values indicated that the normality assumption was

Table 2

Experimental and control groups' descriptive statistics

	Experimental Group					Control Group				
	N	M	SD	Skewness	Kurtosis	N	M	SD	Skewness	Kurtosis
Post-AT	27	53.2	14.4	0.46	-0.65	26	42.5	15.1	0.63	-0.41
Pots-PT	27	50.5	16.7	0.33	-0.73	26	38.0	16.2	0.18	-0.91
Post-PLS	27	34.6	5.3	0.05	-0.05	26	31.7	5.6	0.16	0.35
Post-CPAS Cognitive	27	27.1	1.8	-0.29	-1.00	26	26.5	2.3	0.05	-0.87
Post-CPAS Behavior	27	22.8	3.9	-0.23	0.01	26	21.5	3.6	0.46	0.11
Post-CPAS Affective	27	23.9	4.1	-0.73	0.85	26	23.3	3.5	-0.08	-0.26

Table 3
Results of ANOVA for each dimension

Variable	Group	N	X	Sd	df	F	p
Cognitive	Exp.	27	27.1	1.8	1–51	1.141	0.291
	Cont	26	26.5	2.3			
Behavior	Exp.	27	22.8	3.9	1–51	1.525	0.223
	Cont	26	21.5	3.6			
Affective	Exp.	27	23.9	4.1	1–51	0.308	0.581
	Cont	26	23.3	3.5			

met (Table 2). Box's M was equal to 4.644 and was not significant [f (6, 18767) = 0.724, p > .001]. It showed that equality of covariance matrices assumption has not been violated. Finally, main MANOVA analysis showed that the effect of treatment on students' attitudes toward computer programming was not significant [Wilks Lamda (λ = .968), F (3, 49) = 0.542, p = 0.656].

Univariate ANOVAs were conducted to examine the effect of treatment on dimensions of CPAS as a follow-up to MANOVA. Table 3 shows results of ANOVA for each dimension. Non-significant effect of treatment on students' attitudes on each dimension was found with respect to ANOVA results [Affective: F (1, 51) = 0.308, p > .05; Cognitive: F (1, 51) = 1.141, p > .05; Behavior: F (1, 51) = 1.525, p > .05].

Qualitative Results

The aim of the qualitative phase of the study was to shed more light on the quantitative findings. Qualitative data analysis produced results related to understandings of the participants related to loops concept. As a result of qualitative data analysis three main categories were determined. These categories were called:

(i) Action limited.
(ii) Simultaneous nested loop.
(iii) Object matured.

Action Limited

This is a kind of low level understanding in which individuals cannot imagine instructions in the loop body being performed and they need to explicitly perform the instructions. This category will be called Action limited. It was found that 5 of the 9 control group pre-service teachers 2 of the 9 experimental group pre-service teachers showed indications of Action limited. Pre-service teachers were given the following code segment and asked to determine the output of it.

```
int a[7], b[7], i;
for (i = 7; i >= 1; i-- )
    b[i-1] = i*i;
for (i = 7; i >= 1; i--)
    a[i-1] = i+2;
printf("%d", a[3] + b[4]);
```

This question can be solved by using I = 5 for the first loop that makes b[4] = 25 and then taking i 4 for the second loop that makes a[3] = 6. So the result can be found easily by adding 25 and 6, 6+25 = 31. Nevertheless, as can be seen in the following excerpt, the pre-service teacher insists to find all values till a[3] and b[4] to get the result.

> *Student 8:* i starts from 7. i is 7. Then from 7-1, b[6] is equal to 7x7, 49. For the second loop, i is 7. Then a[6] is equal to, i is 7 from 7+2 it becomes 9… Then it will go to the beginning again. For the first loop i is 6. i-1 is 5 and since i is 6, 6x6=36 [she wrote b[5]=36]. Again for the second loop, i is decreased by one and is is 6. Then a[5] uhmm, from 6+2, is 8. Now i is decreased by 1 and is 5. i-1 is 4 and square of i is 25 [she wrote b[4]=25]. For the second loop i is 5. a[4] is equal to uhmm 7. I have found b[4], I will not enter in the first loop again but continue with the second loop since I need to find a[3]. i was 5 and now i is 4. a[3] is equal to 6, from 4+2. I have found b[4] and a[3], their addition is 31. The output is 31.

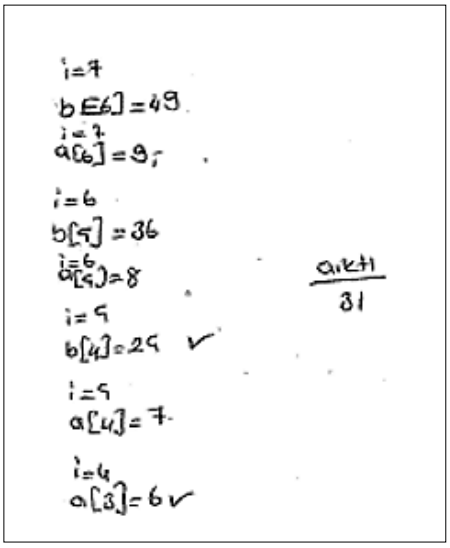

Fig. 2. Pre-service teacher's writings.

Simultaneous Nested Loop

In the literature it was found that some students from different grade levels show a cognitive difficulty in understanding nested loops (Izu, Weerasinghe, and Pope, 2016; Mladenovic, Boljat and Zanko, 2018). They stepped through simultaneously control variables of a nested loop, i.e. they run the inner and outer loops of a nested loop simultaneously. This misconception will be called simultaneous nested loop in this study. Three of the control group pre-service teachers showed simultaneous nested loop misconception while only one of the experimental group pre-service teachers showed the misconception. Pre-service teachers were given the following code segment and asked to find the value of x after the code executed.

```
int i, j, x = 0;
for(i = 1; i <= 9; i++){
    for(j = 1; j <= 20-2*i; j++){
        x = x+i;
    }
}
```

The following excerpt can be given as an example for simultaneous loop misconception. The pre-service student determined a value for the control variable of inner loop for each value of control variable of the outer loop. Except the first one (for i equals 1) she took the upper limit of j for each i. Then two loops were run simultaneously for the values of i and j. When the test condition of the outer loop became false, she stopped the simultaneous iteration.

> *Student 3:* For i equals to 1, the value of j is 1. x was 0 at the beginning. From 0+1, x is 1. For i equals to 2, the value of j is 16. x was 1 and we will add i to it. x is 3 now. For i equals to 3, the value of j is 14. x was 3. I add 3 to 3. It is 6. For i equals to 4, the value of j is 12. x was 6. 6 plus 4, x is equal to 10. For i equals to 5, the value of j is 10. From 10+5, x is 15. For i equals to 6, the value of j is [few-second silence] 8. From 15+6, x is 21. Then for i equals to 7, the value of j is [few-second silence] 6. From 21+7, x is 28. For i equals to 8, the value of j is [few-second silence] 4. From 28+8, x is 36. For i equals to 9, the value of j is 2. From 36+9, x is 45. Since i should be less than or equal to 9, it stops here and the value of x is 45.

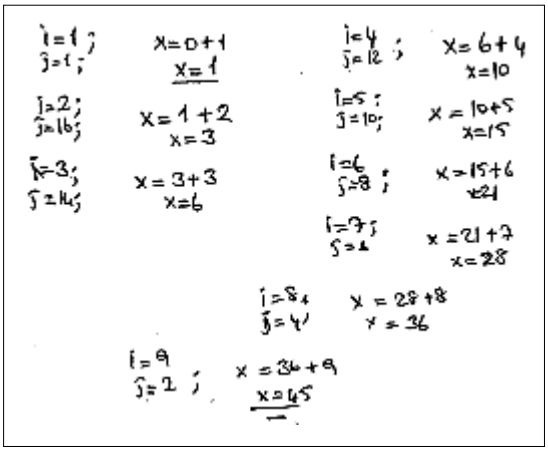

Fig. 3. Pre-service teacher's writings

Object Matured

Cetin (2015) indicated that there is a 'mature' kind of understanding of loop concept called Object conception. Individuals having this conception can see a loop as a whole and can apply operations on it. It can be considered as a procedure or function with input (boundaries of control variable), process (loop is run) and output (afterward if any). At this stage, loop is an object, and students can consider it as a single command. The category in which pre-service teachers showed indications of Object conception will be called Object matured. Two of the experimental group pre-service teachers were classified under Object matured while none of the pre-service teachers in the control group was classified under Object matured. Pre-service teachers were given the following code segment and asked to find output of it.

```
int i, j, k;
for (i = 1;i <= 10; i++){
    for (j = 1; j <= 5;
j++){
        for(k = 1; k <=8;
k++){
            printf("*");
        }
    }
}
```

The following excerpt can be given as an example for Object matured category. The pre-service teacher saw the inner most loop as 8 stars, in other words he showed indications of seeing the inner most loop as a totality. He was aware that middle loop repeats printing 8 stars 5 times that results in 40 stars. This is also an indication for totality. This is especially evident when he said "Assume there is one nested loop including two fors". Then he multiplied 10 by 40 to get the result that means multiplication operation (or one can also say repetition) is applied on the totality.

> *Student 10:* At the first stage this will print 8 stars [he means asterisk], then it moves to the upper for. For j equals to 2, it again prints 8 stars. Rather than proceeding this way for each one, we can get the solution by multiplying these three [he means 10*5*8].
>
> *Interviewer:* Why will this give the solution?
>
> *Student 10:* Assume there is one nested loop including two fors, assume there are j and k and printf("*") inside. At the first step, this [he showed innermost for loop] will run 8 times, then j becomes 2 and this will run again 8 times, that is this prints 8 stars and then 8 stars. This is done 5 times. From 8*5 we get 40. Then it will move to one upper, moves to i for-loop. Then for i equals to 1 we have 40 stars, for i equals to 2 we have 40 stars. Since i equals to 10, by multiplying 10 by 40 we get 400 stars.

Discussion

Teacher candidates in the experimental group performed significantly better than the teacher candidates in the control group with respect to both AT and PT. AT requires pre-service teachers to both understand given code and solve the question depending on their understanding. In addition to this, pre-service teachers needed to construct their own program to solve the given questions in PT. The reasons for significantly better performance of pre-service teachers in experimental group can be considered from different perspectives. One such apparent reason in visualization research is related with the engagement levels of the participants. The pre-service teachers in experimental group had a more active role in their learning. So, it can be said that their active engagement helped them to improve their understanding (Hundhausen *et al.*, 2002; Naps *et al.*, 2003). Active engagement is a plausible explanation but it needs a closer consideration to get a better explanation of the phenomena.

The intervention given in experimental group depended mainly on Scratch based visualization and V-A model of Zazkis, Dubinsky and Dautermann (1996). The reason of students' better performance might be due to these two components (or their combination) in addition to active engagement explanation. Scratch is a high ceiling, wide walls, and low floor, programming environment (Resnick *et al.*, 2009). It can be said that use of Scratch might have a positive effect in helping students to learn. Pre-service teachers

needed to use visual loop blocks present in Scratch to learn the loops concept in a text-based environment. The use of block based loops is easier and can form a step to learn harder text based loops. This will be called dual effect.

The second component of the intervention was V-A model. The V-A model posits that visualization helps individuals to improve their analytic understanding (Zazkis, Dubinsky and Dautermann, 1996). Depending on the V-A model one can hypothesize that pre-service teachers who were given loops instruction depending on the V-A model have more coherent understanding and less misconceptions. The qualitative findings of the study is in line with this hypothesis. Fewer pre-service teachers who were interviewed showed simultaneous nested loops misconception in the experimental group. Fewer pre-service teachers who were interviewed showed low level understanding of loops concept (called Action limited) in the experimental group. More pre-service teachers who were interviewed showed a coherent understanding of loops concept (called Object matured) in the experimental group.

No significant difference between the experimental and the control group with respect to perceived learning was found, although experimental group pre-service teachers' mean score was higher than the control group pre-service teachers' mean score. Similarly, no significant difference between control and experimental groups in terms of their programming attitudes was found. It might be concluded that the intervention failed to produce intended improvement in experimental group pre-service teachers' perceived learning and attitudes in the context of the study. However, this result should be carefully read considering the duration of the study. The intervention lasted in three weeks in the study. Three weeks might not be enough to observe attitude difference (Papanastasiou and Papanastasiou, 2004). Results of the study related to perceived learning supports this idea. Perceived learning might play an intermediary role between programming achievement and attitude in the context of this study. Significant difference was detected between experimental and control group in terms of programming achievement. However, non-significant difference was detected in terms of perceived learning. The non-significant difference might be due to the fact that pre-service teachers could not construct the awareness of their achievement in three weeks yet. This might result in no difference between experimental and control groups in terms of their programming attitude. This idea can be further tested by using structural equation modeling in the future studies.

Conclusion

There have been an interesting phenomenon in visualization research. The computing teachers or instructors stated that they believe visualization is useful in teaching their courses but at the same time they are not willing to use it in their courses (Baker and O'Neil Jr, 2002; Shaffer *et al.*, 2010). It might be the case that integration of an additional visualization tool that is hard to learn and will not be used for programming (the main purpose of the course) is not practical for them. Visual programming environments like Scratch can be a solution (Xu, Ritzhaupt, Tian and Umapathy, 2019). They

are easy to learn and individuals practice computing while constructing visualizations to learn the concepts of computing which is called dual effect in the study. Teachers or instructors experiencing the dual effect might be more willing to incorporate visualization in their courses. They can experience dual effect both in (i) learning the introductory programming concepts first time in their university years and (ii) their teaching in their classes after graduation.

The main framework in visualization research is related with students' role in visualization activities. Hundhausen, Douglas and Stasko (2002) figured out students playing more active role in visualization activities gain more understanding in programming. Following this finding Naps *et al.* (2003) proposed an engagement taxonomy to classify and take attention to students' role in visualization activities. It can be said that learning environment that does not require active involvement of learners would probably fail to help students constructing or improving their understanding. Although active engagement is important in helping learners to improve their learning, it cannot be responsible for all the success in helping students to improve their learning. Cetin (2013) found and argued that a specific type of active engagement caused different results in helping students to improve their understanding. One can construct different kinds of instructions in which learners can play different active involvement roles (Mikropoulos and Bellou, 2013). Moreover, it is possible that these roles might suit better for some situations. The idea that knowledge is actively constructed by the learner or learners should have active involvement in their learning is called trivial constructivism (Von Glasersfeld, 1991). Trivial constructivism falls short in explaining learning. Therefore there is a need for frameworks or theories to situate visualization in a broader context to explain learners' understanding and construct instruction to help learners to improve their understanding. Al-Sakkaf, Omar, and Ahmad (2019) conducted a review study and stated that there is a need for theories, models, or frameworks in visualization research. In the current study, V-A model was used to construct the visualization activities. It can be concluded from the findings of the study that V-A model was useful in helping students to improve their understanding. The V-A model sees visualization as a tool to help students construct necessary abstractions. Having necessary abstractions might help pre-service teachers develop better schema and have a good base for problem solving related to concept (Dubinsky, 1991).

Block based programming environments can enhance teachers' daily classroom practice. They are easy to learn and students can tinker to explore basics of programming. However, these are the property of the programming environment, its pedagogical value will be created by teachers as designers of the instruction. Abstraction is one of the keys or barriers in the learning programming and developing computational thinking. Students need to construct necessary abstractions to achieve in computer science classes. Block based programming environments in the visualization context can be used to create dual effect (being both tool and content for abstraction) for students. Students used more concrete concept of loops in block based visualization context to learn more abstract concept of loops in C in this study. As a (relatively) concrete visualization tool, with which students developed virtually concrete products, block based programming environments can help students develop necessary abstractions. In addition to this, stu-

dents already started to learn the loop concept while involving in visualization in this study, i.e. the visualization tool was itself the content of the study. Teachers can produce practical pedagogical value from the property of blocked based programming environments in the visualization context. The dual effect can be utilized in the instruction by teachers for students' concept formation process.

The intervention in the study lasted three weeks. Three weeks of instruction related to loops might not be enough to master loops concept. The preservice teachers would see the concept throughout their programming courses again and again. This study is limited to three weeks part of preservice teachers' whole experience. In addition to this, visualization was add on to the classroom instruction. This might be counted as limitation. Whole instruction could be designed based on visualization. However, this approach might have its disadvantages too. The teachers/instructors need to radically change their usual way of teaching. They might be less willing to adapt radical changes. Future studies can explore this hypothesis. Moreover, the study included 53 participants. Future studies can be done with more participants.

References

Al-Sakkaf, A., Omar, M., & Ahmad, M. (2019). A systematic literature review of student engagement in software visualization: a theoretical perspective. *Computer Science Education,* 1–27.

Baker, E. L., & O'Neil Jr, H. F. (2002). Measuring problem solving in computer environments: Current and future states. *Computers in Human Behavior, 18*(6), 609–622.

Ben-Ari, M., Bednarik, R., Levy, R. B. B.,, Ebel, G., Moreno, A., Myller, N., & Sutinen, E. (2011). A decade of research and development on program animation: The Jeliot experience. *Journal of Visual Languages & Computing, 22*(5), 375–384.

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.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*. Vancouver, Canada.

Cetin, I. (2013). Visualization: a tool for enhancing students' concept images of basic object-oriented concepts. *Computer Science Education, 23*(1), 1–23.

Cetin, I. (2015). Students' understanding of loops and nested loops in computer programming: An APOS theory perspective. *Canadian Journal of Science, Mathematics and Technology Education, 15*(2), 155–170.

Creswell, J. W., & Clark, V. L. P. (2007). *Designing and conducting mixed methods research*. Thousand Oaks, CA: Sage.

Dubinsky, E. (1991). Reflective abstraction in advanced mathematical thinking. In D. Tall (Ed.), Advanced mathematical thinking (pp. 95–126). Boston, MA: Kluwer.

DuBoulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research, 2*(1), 57–73.

Ginat, D. (2004). On novice loop boundaries and range conceptions. *Computer Science Education, 14*(3), 165–181.

Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing, 13*(3), 259–290.

Izu, C., Weerasinghe, A., & Pope, C. (2016). A study of code design skills in novice programmers using the SOLO taxonomy. *Proceedings of the 2016 ACM Conference on International Computing Education Research* (pp. 251–259). Melbourne, VIC, Australia.

Johnson, R. B., Onwuegbuzie, A. J. (2004) Mixed methods research: A research paradigm whose time has come. Educational Researcher 33(7): 14–26.

Kordaki, M., Miatidis, M., & Kapsampelis, G. (2008). A computer environment for the learning of sorting algorithms: Design and pilot evaluation. *Computers & Education, 51,* 708–723.

Merino, L., Ghafari, M., Anslow, C., & Nierstrasz, O. (2018). A systematic literature review of software visualization evaluation. *Journal of Systems and Software, 144*, 165–180.

Mikropoulos, T. A., & Bellou, I. (2013). Educational robotics as mindtools. *Themes in Science and Technology Education*, *6*(1), 5–14.

Mladenovic, M., Boljat, I., & Zanko, Z. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies, 23*(4), 1483–1500.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., … Velazquez-Iturbide, J. A. (2003). Exploring the role of visualization and engagement in computer science education. *ACM SIGCSE Bulletin, 35*, 131–152.

Papanastasiou, C., & Papanastasiou, E. C. (2004). Major influences on attitudes toward science. *Educational research and Evaluation, 10*(3), 239–257.

Price, B., Baecker, R. M., & Small, I. (1998). An introduction to software visualization. In J. Stasko, J. Domingue, M. Brown, & B. Price (Eds.), *Software visualization: Programming as a multimedia experience* (pp. 3–27). Cambridge, MA: MIT Press.

Putnam, R. T., Sleeman, D., Baxter, J. A., & Kuspa, L. K. (1986). A summary of misconceptions of high school Basic programmers. *Journal of Educational Computing Research, 2*(4), 459–472.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. B. (2009). Scratch: Programming for all. *Commun. Acm, 52*(11), 60–67.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137–172.

Rovai, A. P., Wighting, M. J., Baker, J. D., & Grooms, L. D. (2009). Development of an instrument to measure perceived cognitive, affective, and psychomotor learning in traditional and virtual classroom higher education settings. *The Internet and Higher Education, 12*(1), 7–13.

Schreier, M. (2012). Qualitative content analysis in practice. Thousand Oaks, CA: Sage.

Seidman, R. H. (1989). Computer programming and logical reasoning: Unintended cognitive effects. *Journal of Educational Technology Systems, 18*(2), 123–141.

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*, 10(3): 1–22.

Sivasakthi, M., & Rajendran, R. (2011). Learning difficulties of'object-oriented programming paradigm using Java': students' perspective. *Indian Journal of Science and Technology, 4*(8), 983–985.

Sorva, J., Karavirta, V., & Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE), 13*(4), 1–64.

Sorva, J., Lönnberg, J., & Malmi, L. (2013). Students' ways of experiencing visual program simulation. *Computer Science Education, 23*(3), 207–238.

Stasko, J., Kehoe, C., & Taylor, A. (2001). Rethinking the evaluation of algorithm animations as learning aids: An observational study. *International Journal of Human Computer Studies, 54*(2), 265–284.

Teague, D., & Lister, R. (2014). Programming: reading, writing and reversing. *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 285–290). Uppsala, Sweden: Uppsala University.

Top, E., Yukselturk, E., & Inan, F. A. (2010). Reconsidering usage of blogging in preservice teacher education courses. The Internet and Higher Education, 13(4), 214–217.

Velazquez-Iturbide, J. A., & Perez-Carrasco, A. (2016). How to use the SRec visualization system in programming and algorithm courses. *ACM Inroads, 7*(3), 42–49.

Von Glasersfeld, E. (1991). An exposition of constructivism: why some like it radical. In G.J. Klir (Ed.), *Facets of system science* (pp. 229–238). New York/London: Plenum Press.

Wiedenbeck, S., Ramalingam, V., Sarasamma, S., & Corritore, C. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting With Computers, 11*(3), 255–282.

Wing, J. M. (2011, February). Research notebook: Computational thinking- what and why? The Link Magazine, 20–23. Retrieved from https://www.scs.cmu.edu/link

Xu, Z., Ritzhaupt, A. D., Tian, F., & Umapathy, K. (2019). Block-based versus text-based programming environments on novice student learning outcomes: a meta-analysis study. *Computer Science Education*, 1–28.

Zazkis, R., Dubinsky, E., & Dautermann, J. (1996). Coordinating visual and analytic strategies: A study of students' understanding of the group D 4. *Journal for research in Mathematics Education*, 435–457.

I. Cetin received his PhD degree in Computer Education and Instructional Technologies Department in Middle East Technical University, Ankara, Turkey in 2009. Currently he is an associated professor in Computer Education and Instructional Technology Department in Bolu Abant Izzet Baysal University. He is interested in mathematical thinking, computational thinking, the link between mathematical and computational thinking, and the use of technology (e.g. computer programming, visualization, games and computer algebra systems) in computer science and mathematics education.