*I.J. Modern Education and Computer Science*, **2016, 1, 16-23**
Published Online January 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2016.01.03

# The Effects of Scratch-Based Game Activities on Students' Attitudes, Self-Efficacy and Academic Achievement

**Özgen KORKMAZ**
Amasya University, Faculty of Technology, Department of Computer Engineering, Amasya, Turkey
E-mail: ozgenkorkmaz@gmail.com

***Abstract*—**The purpose of the present research is to designate the effects of Scratch-based game activities on students' attitudes towards learning computer programming, self-efficacy beliefs and levels of academic achievement. The research was conducted through a pre-test – post-test control group quasi-experimental study. The study group consists of 49 students studying at the Faculty of Engineering. The test group was administered a teaching method with Scratch-based game activities. On the other hand, the control group was directly taught C++ topics via an editor. Research data were collected via an implementing academic achievement test (Kr-20= 0, 71), attitude towards a learning programming scale (Cronbach's Alpha =0.84) and a computer programming self-efficacy scale (Cronbach's Alpha= 0.966). Our findings are as follows: A significant number of the students consider themselves as mid-level efficient in C++ programming. Scratch-based game activities render no effects on students' attitudes and self-efficacy perceptions. On the other hand, Scratch-based game activities render significant contributions on students' academic achievements in C++ programming language.

***Index Terms*—**Scratch, programming, learning, learning environment, Scratch - based game.

## I. Introduction

As stated in the various studies, in order for individuals to do well in computer programming, the prerequisite higher-order thinking skills are problem solving, logical and mathematical thinking, critical thinking and creative thinking [1 – 5]. Likewise, Hamada [6] argues that one of the ways to polish such higher-order thinking skills is rendering education on computer programming. It has been manifested that programming education rendered to students at an early age plays an effective role in gaining skills such as mathematical thinking and problem solving [quot: 7]. Akpınar and Altun [8] claim that by virtue of the various contributions of programming education to students, it should not be limited to undergraduate education but encompass elementary and secondary level education as well. Similarly, setting the curriculum on the basis of this program and design would, they argue, enhance students' analytical and spatial thinking skills and problem solving skills. In this respect, it would be unwise to label programming education as a vocational training. Rather, it should be deemed, just like mathematics, as an alternative path to gain proper thinking skills throughout one's academic life. From this perspective, the issue of programming education gains further momentum on account of its capacity to develop necessary skills for students from all walks of educational stages.

In the relevant literature, it is feasible to come across a range of studies on the challenges experienced while teaching and learning computer programming [1, 3, 9-14]. The challenges experienced may be attributed to several factors, some of which are the fact that individuals fail to possess the above-mentioned higher-order thinking skills, apply incomplete or improper teaching approaches, or attribute poor significance to computer programming education [3, 4, 13, 15].

In particular, the formation of programming language with abstract statements and the failure to grasp the link between abstract statements and consequential behaviours may mistakenly lead learners to perceive the programming process as an abstract, even unreal phenomenon from the very beginning to the end. Recently it has become possible to create a number of environments in which programming structures can be used visually and the results of the created program can be monitored tangibly. Scratch, Logo and Small talk are among such environments. Scratch was developed within the scope of a project that was executed by MIT between 2003-2007 and supported by the National Science Foundation (NSF). The objective of this project was, as reported, to boost under-educated students from less developed social groups to those who exhibit better skills in technology use which would, in effect, enable them to create meaningful products by making use of advanced technologies [16, 17]. Scratch is a recommended programming language to gain algorithmic thinking skills for those who freshly start programming education [18, 19]. Resnick et al. [16] and Kaučič & Asič [20] claim that computer programming via Scratch may make it easy for all to understand as a favour of the basic design features of the Scratch environment. It is possible for students to create program components by dragging and dropping graphical block structures. Once dissimilar commands are

ordered one under the other exactly and inter-fitting geometric shapes which alleviate students' tendency to commit spelling mistakes and memorize command scripts are created. Most activities in traditional programming education are built upon unenjoyably numeric, string and simple graphical processes [21]. In reality, however, the new programming environment is conducive to perform processes on the new images, animations, film parts and sounds [17]. Aside from that, young people can share their projects with their peers on the web and exchange opinions. Moreover, it can support a number of natural languages to make it easier for speakers of other languages to communicate and cooperate. The Scratch environment supports 42 language options. Scratch is a free application easily downloadable from its original web address or can be used online without being downloaded. Since downloaded projects on the Scratch web site are open source coded, any user can download and examine the project and establish communication with no language barriers among other users while interacting and devising new projects [22].

In the relevant literature, an extensive body of research exists on Scratch. Genç and Karakuş [23] in their study stated that a majority of Scratch-user students reported that Scratch was simple, enjoyable and convenient to use. In his research, Calder [24] claimed that Scratch rendered contributions to the development of mathematical thinking, problem solving, logic improvement and analytical thinking skills. Hence, he suggested using the Scratch program education to students at an early age. In a similar study conducted by Kaučič and Asič [20], it was stated that the Scratch application supported the development of problem solving and algorithmic skills in children. There are many literature studies echoing that Scratch has positive effects on students' programming and basic thinking skills [18, 20, 23]. On the other hand, in the relevant literature no studies have been identified indicating the concrete outcomes of offering the Scratch education so as late as undergraduate level. To put it in another way, the effectiveness of providing students education with the logic of basic programming via Scratch before giving them C++education is still a matter of curiosity. Within this framework, the purpose of the present study is to designate the effects of Scratch-based game activities on students' attitudes towards learning computer programming, self-efficacy beliefs and levels of academic achievement.

In the relevant literature, large quantities of evidence exist on the direct effects of the attitudes of students towards school, lessons or teachers on the basis of a set of variables such as academic achievement, self-efficacy perceptions and fulfilment [25, 26]. Negative perception, motivation and attitude are, as reported, particularly stronger elements when compared to the other factors affecting academic achievement [27, 28]. Self-efficacy is defined as one's inner faith in his/her ability to succeed in any given task. This faith determines one's willingness to attempt a particular behaviour, to continue this behaviour, to keep motivated and as a result, the final outcome in the form of academic achievement [29]. On that account, in the present research, not only academic achievement but also attitude and self-efficacy perceptions towards programming education have also been dealt with.

***Problem Statement:***

Do Scratch-based game activities have any effects on students' attitudes towards learning computer programming, self-efficacy beliefs and levels of academic achievement?

***Sub-problems:***

a On a general scale, what are the students' academic achievements and attitudes towards learning computer programming and programming self-efficacy beliefs?
b Before the application, are the groups similar in terms of academic achievements and attitudes towards learning computer programming and programming self-efficacy beliefs?
c Do traditional methods and Scratch-based game activities change students' academic achievements towards programming and attitudes towards learning programming and self-efficacy beliefs?

## II. Methods

***Research Design***

The research consists of a pre-test-final test control group quasi-experimental study. The graphic image of the test model employed in the research is as follows:

Table1. Experimental Design

| Groups | Pretest | Test Manipulation | Posttest |
|---|---|---|---|
| Test group | • Academic Achievement Test<br>• Attitude towards learning programming scale<br>• Computer programming self-efficacy scale | Scratch-based game activities | • Academic Achievement Test<br>• Attitude towards learning programming scale<br>• Computer programming self-efficacy scale |
| Control Group | | Traditional Method | |

***Study Group***

The study group of this research is composed of collectively 55 students. Of the study group taking 3-credit Computer Programming course at the Faculty of Engineering Department of Computer Engineering and Electric – Electronics Engineering Department 1st grade, 14 students are female and 41 are male in both sections. While creating the test groups, no intervention was made to the groups and natural classroom structures in schools were kept the same. A random selection method was administered in designating the groups as either test or

control groups. Nonetheless, 6 students who failed to take the pretest or the final test, or who failed to continue the practices were omitted from the evaluation. When unevaluated participants were excluded, the distribution of study groups with respect to gender was summarized in Table 2.

Table 2. Distribution of Study Groups with Respect to Gender

| Groups | Male | Female | Total |
|---|---|---|---|
| **Test group** | 16 | 5 | 21 |
| **Control Group** | 21 | 7 | 28 |
| **Total** | 37 | 12 | 49 |

### *Test group*

The test group was administered a teaching method with a focus on Scratch-based game activities. Accordingly, for a period of two weeks students were introduced to the Scratch environment in which all programming elements were exemplified via samples. Until the last week of the application, the students were asked to design an error-free game by using Scratch in such a way that the game should contain minimum two characters, two interactions and two displays. The reason why these activities in the control group are called an instruction based on Scratch-based play activities is that the expectation is learning during teaching should be realized while designing play activities. Following the Scratch sessions, lectures were provided on variables in C++, basic structuring of C++, program controls, loops, conditions, functions and the basic available functions for a period of six weeks and the samples were shared via Scratch as well as C++editor. Each week, the students were given feedbacks and corrections on the games as they kept designing in the Scratch environment. At the end of the six-week period, they were given a game contest on their game tasks allocated at the start of the application and the games were evaluated by the students and the researchers with respect to the final product and programming stages. The highest-ranking three students were given some minor awards. Some sample images from the games of students are given in Fig 1.

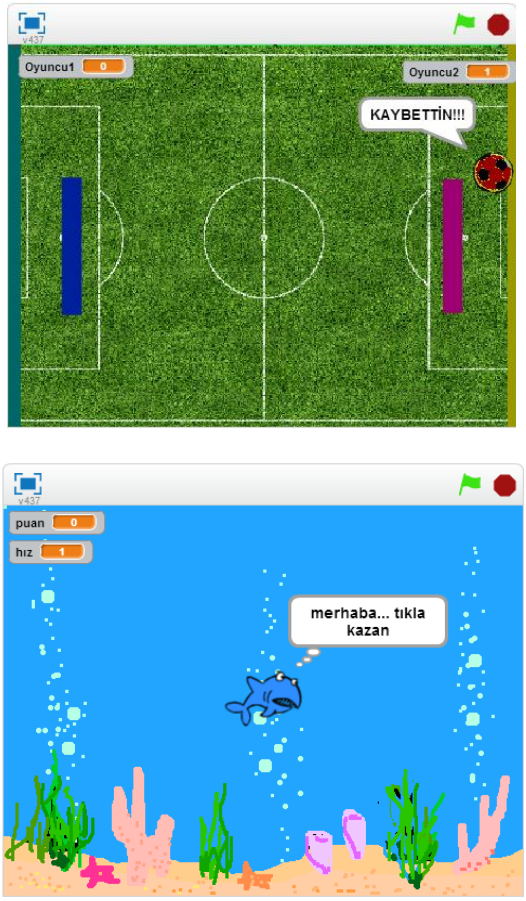



Fig.1. Images of Sample Games

### *Control Group*

In the control group, pre-test and final test practices were administered as identical to the first and last weeks. In the remaining eight-week period, the very same C++ subjects were lectured. Upon exemplifying each subject, C++editor was used to solve the sample cases. Also, the students were given homework assignments and before all lesson assignments were checked in class. Unsolved questions were explained in class. The test process tracked in both test and control group is as listed in Table 3.

Table 3. Test Processes Practiced in Test and Control Groups

| Week | Subject | Test group | Control Group |
|---|---|---|---|
| 1 | Administering pretests and rendering extensive information on the process | X | X |
| 2 | General structure of Scratch | X | |
| 3 | General structure of Scratch | X | |
| 4 | Background of C; Main Structure of C; Operators; Priority operators; Data Types; | X | X |
| 5 | Assignment and Entry Exit commands; Printf; Scanf | X | X |
| 6 | Select (flow) commands; If-Else/Switch | X | X |
| 7 | Loop commands; While/for/do while/ break/ continue | X | X |
| 8 | Functions/Indicators | X | X |
| 9 | Sequences /Strings | X | X |
| 10 | Scratch-Based game contests | X | |
| 11 | Final Test Applications | X | X |

### *Data Collection Tools*

*Academic Achievement Test:* An academic achievement test was formed within the scope of the present research. 30 items with 5 multiple-choice questions which contain topics such as variables in C++, basic structure of C++, program controls, loops, conditions, functions and the basic available functions were prepared. As a result of a pilot study conducted on 148 students who had previously taken a C++ course 5 items, the distinctiveness level of which was below 0,3 and the exclusion of which would not disrupt content validity were removed from the scale. In its final version, the scale contained 25 items and the distinctiveness coefficients of the items varied between 0.296 and 0.740 and its difficulty index was measured as 0.51. Kr-20 internal consistency coefficient was measured as 0.71.

*Attitude towards the learning programming scale:* The research data were compiled through the "Attitude towards Computer Programming Scale" developed by Korkmaz and Altun [1]. The Attitude towards the Computer Programming Scale was developed by Korkmaz and Altun [1] to detect students' attitudes in terms of validity and reliability. The scale consisted of total 20 items and three factors. Each single item was scaled as; "Never (1), Rarely (2), Sometimes (3), Generally (4), Always (5)". The construct validity and internal consistency information of the scale are summarized in Table 4.

Table 4. Construct Validity and Internal Consistency

| Factors | Items | KMO | Bartlett | Eigenvalue | Explained Variance | Cronbach Alpha |
|---|---|---|---|---|---|---|
| F1 (Motivation) | 9 | 0.876 | $x^2$=2867.942; sd=190; p<0.001 | 5.70 | 17.55 | 0.82 |
| F2 (Negative Attitude) | 6 | | | 2.10 | 16.01 | 0.77 |
| F3 (Necessity) | 5 | | | 1.1 | 13.69 | 0.75 |
| **Total** | **20** | | | **-** | **47.34** | **0.87** |

Table 4 manifests that the KMO value is 0.876 and the Bartlett value is below 0.05. The internal consistency coefficient value of the whole scale 0.87. The factor called motivation, which is one of the items related to the measurement of students' willingness status for learning programming, reveal the willingness levels of students for learning programming. The factor referred to as negative attitude, reveal the negative opinions of students related to learning programming. The factor called necessity includes items related to whether the students think that it is necessary to learn programming or not. Within this framework, it is safe to claim that the scale is a valid and reliable tool in measuring students' attitudes towards learning computer programming.

*Computer programming self-efficacy scale:* In order to designate students' self-efficacy beliefs towards programming, Computer Programming Self-efficacy Scale developed by Ramalingam and Wiedenbeck [30] and adapted into Turkish by Korkmaz, Altun [31] were utilized. In line with the adaptation analyses made by Korkmaz and Altun [31], this scale which was originally composed of 32 items was reduced to 28 items and one factor. The factor loads of scale items varied between 0.618-0.807. the item factor correlations are between 0.621 and 0.804, corrected correlations are between 0.588 and 0.779. the internal consistency coefficient is (Cronbach's Alpha) 0.966. Accordingly, the scale is in Turkish culture as well, a valid and reliable tool to measure students' self-efficacy beliefs towards computer programming.

### *Data Analysis*

The data collection tools employed in the research fail to be standard due to the scores corresponding to the responses given to 5, and 7-Likert type scales and differentiations in the number of items in the subfactors. Therefore, the best method seemed to be to convert the obtained raw scores into the standard scores, the lowest of which would be 20, and the highest of which would be 100. Accordingly, the standardized score were examined by using frequency, percentage, arithmetic means, standard deviation and t analyses. On the other hand, students with a score of 46 and lower were categorized in the lower level, between 47 and 72 as average level and the ones with 72 and higher scores as in higher-order category.

## III. FINDINGS

### *Students' Academic Achievements, Attitudes towards Learning Programming and Programming Self-Efficacy Beliefs*

On a broad sense, students' academic achievements in programming, attitudes towards learning programming and self-efficacy beliefs were examined with respect to their final test scores. Accordingly, the findings on academic achievement are listed in Table 5.

Table 5. Students' Academic Achievements

| Acad. Achiev. | N | Min | Max | Mean | Std. Dev. | Skew. | Kur. |
|---|---|---|---|---|---|---|---|
| **Test Group** | 28 | 8.00 | 80.00 | 54.8 | 15.8 | - | - |
| **Cont. Group** | 21 | 20.00 | 88.00 | 44.9 | 17.5 | - | - |
| **Total** | 49 | 8.00 | 88.00 | 49.1 | 17.1 | 0.069 | 0.226 |

In Table 5, as seen from the academic achievement final test scores of the students, out of 100 total scores, the lowest score that the students can get is 8, and the highest score is 88. The academic achievement mean of students is 49.1. Thus, it can be claimed that the students' C++ programming academic achievement is quite low on the whole. This might be related to the fact that the students were not adequately supported to improve their higher-order thinking skills. On the other hand, as Skewness and Kurtosis figures are examined, both numbers received values between -1.5 and +1.5, which means that their academic achievement scores are within the normal distribution range (Shapiro-Wilk$_{49}$=0.971, p>0.05) [32]. The students' attitudes towards learning programming and their self-efficacy perception levels are listed as the findings under Table 6.

Table 6. Students' Attitudes towards Learning Programming and their Self-Efficacy Perceptions

| Variables | N | Min | Max | Mean | Std. Dev. | Skew. | Kur. | Levels(f/%) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Low | Med. | High |
| **Self-Efficacy** | 49 | 33 | 96 | 62.6 | 16.7 | .34 | -.63 | 11 %22.4 | 24 %49.0 | 14 %28.6 |
| **Motivation** | | 24 | 100 | 72.7 | 17.4 | -.42 | -.01 | 3 %6.1 | 21 %42.9 | 25 %51.0 |
| **Negative Attitude** | | 47 | 100 | 77.8 | 17.9 | -.28 | -1.17 | 3 %6.1 | 12 %24.5 | 34 %69.4 |
| **Necessity** | | 20 | 76 | 32.2 | 14.9 | 1.42 | 1.41 | 43 %87.8 | 5 %10.2 | 1 %2 |

Table 6 reveals that the students' self-efficacy perceptions towards programming change between 33 and 96 and the mean score is 62.6. On the other hand, 22.4% of the students had low self-efficacy beliefs and 49% had mid-level self-efficacy beliefs while 28.6% had high self-efficacy beliefs. Thus, it is safe to claim that a majority of the students consider themselves mid-level efficient in C++programming.

As the attitudes towards learning C++ programming are examined, it can be said that in the motivation factor the lowest score is 24 and the highest score is 100. The mean score is 72.7. On the other hand, 51% of the students are highly motivated towards learning C++ programming and 42.9% are mid-level motivated. Only 6.1% of all the students lacked motivation. In the negative perception factor, the lowest score is 47 and the highest score is 100. The mean score is 77.8. On the other hand, 69.4% of students have high level of negative perception towards learning C++ programming and 24.5% have a mid-level negative perception. In the necessity factor, the lowest score is 20 and the highest score is 76. The mean score is 32.2. On the other hand, 87.8% of students consider that C++ programming learning is a little necessary, 10,2% consider that C++programming learning is quite necessary. These findings evidence that a great majority of the students are highly motivated to learn C++, but a vast number of the students have negative thoughts on learning C++. It is also true that most of the students consider learning C++ unnecessary.

On the other hand, in terms of both self-efficacy and attitude factors, the scores are distributed in a normal range.

### *Findings as regards the Equivalence of Groups prior to Application*

In Table 7, it was investigated whether the groups were equal in terms of both their academic achievement and their attitudes towards learning programming and level of efficacy prior to the application.

Table 7. Findings as Regards the Equivalence of Groups

| Variables | | N | $\bar{X}$ | S.S | t | df | p |
|---|---|---|---|---|---|---|---|
| F1 (Motivation) | Test Gr. | 28 | 69.7 | 13.9 | 0.833 | 47 | 0.41 |
| | Control Gr | 21 | 73.1 | 13.7 | | | |
| F2(Negative Attitude) | Test Gr. | 28 | 71.6 | 15.1 | 0.140 | | 0.89 |
| | Control Gr | 21 | 72.2 | 15.9 | | | |
| F3(Necessity) | Test Gr. | 28 | 30.8 | 13.2 | -0.531 | | 0.60 |
| | Control Gr | 21 | 28.8 | 12.8 | | | |
| Academic achievements | Test Gr. | 28 | 27.4 | 12.4 | 0.714 | | 0.48 |
| | Control Gr | 21 | 29.9 | 11.3 | | | |
| Self-efficacy | Test Gr. | 28 | 41.4 | 18.4 | 1.307 | | 0.19 |
| | Control Gr | 21 | 35.6 | 10.1 | | | |

Table 7 shows that in both tests and control groups, despite the minor differences in terms of attitudes towards learning programming (motivation, negative attitude, necessity) and academic achievements in C++programming and self-efficacy perceptions, the differences are not significant. It can thus be argued that prior to the application both groups were equal in terms of academic achievement, attitude and self-efficacy.

Although the differences among groups were not insignificant, the scores for the final test and pre-test difference were also used in the analyses given below to be able to control the effects of the designated minor differences as well.

### *Effects of the Scratch-based game activities*

Findings on whether any differences existed with respect to the applied test method and traditional method as regards their academic achievement and attitudes are summarized in Table 8.

Table 8. Effectiveness of Test Application

| Variables | | N | $\bar{X}$ | S.S | t | df | p |
|---|---|---|---|---|---|---|---|
| F1 (Motivation) | Test Gr. | 28 | -.57 | 25.68 | -0.122 | 47 | 0.90 |
| | Control Gr | 21 | -1.39 | 21.27 | | | |
| F2(Negative Attitude) | Test Gr. | 28 | 3.86 | 13.98 | 0.763 | | 0.45 |
| | Control Gr | 21 | 7.36 | 17.17 | | | |
| F3(Necessity) | Test Gr. | 28 | .762 | 13.06 | 0.695 | | 0.49 |
| | Control Gr | 21 | 3.86 | 16.95 | | | |
| Academic achievements | Test Gr. | 28 | 27.43 | 16.31 | -2.636 | | 0.01 |
| | Control Gr | 21 | 15.00 | 16.35 | | | |
| Self-efficacy | Test Gr. | 28 | 24.0 | 19.7 | 0.128 | | 0.89 |
| | Control Gr | 21 | 23.28 | 18.8 | | | |

Table 8 indicates that compared to the traditional method, the Scratch-based game activities created no significant differentiation in terms of motivation ($t_{(2-47)}$=-0.122; p>0.05) or negative attitude ($t_{(2-47)}$=0.763; p>0.05) or necessity ($t_{(2-47)}$=-2.636; p>0.05) dimensions on students' attitudes towards learning programming. It can thus be argued that teaching via Scratch-Based games, in comparison to the traditional method, has no effects on students' attitudes towards programming education. As students' final test attitude mean scores are examined, the mean of motivation factor was 72.6, the mean of the negative attitude score was 0.77 and the mean score of necessity factor was 32.2. So, despite the high level of motivation factor, most of the students consider programming education unnecessary and exhibit quite high levels of negative attitudes. In that case, the frequent negativity in the attitudes of both test and control group students might be the reason why there was no significant differentiation. Table 8 demonstrates that the Scratch-based game activities, unlike the traditional method, significantly differentiates students' academic achievements in C++ ($t_{(2-47)}$=-2.636; p<0.05). The mean score of the test group in academic achievement is 27.43 while the same mean score in the control group is 15.00 and the significant differentiation is in favor of the test group. It is thus safe to argue that teaching via the Scratch-based game activities can, compared to the traditional method, significantly contribute to students' academic achievements in C++ programming language.

Table 8 shows that the Scratch-based game activities, unlike the traditional method, do not differentiate self-efficacy perceptions towards C++ ($t_{(2-47)}$=0.128; p<0.05). Consequently, teaching via the Scratch-based game activities, unlike the traditional method, has no effects on students' self-efficacy beliefs towards C++ programming language.

## IV. Conclusions and Discussion

A significant percentage of the students consider themselves mid-level efficient in C++ programming. It can be claimed that a majority of the students are highly motivated to learn C++ while a significant portion of the students also have negative thoughts towards learning C++. At the same time, a considerable number of these students think that learning C++ is not necessary.

Compared to the traditional method, education via the Scratch-based game activities has no effects on students' attitudes towards programming education. It was also found out that, compared to the traditional method, education via the Scratch-based game activities has no effect on students' self-efficacy towards C++ programming language. Despite the factor of high level of motivation, most of the students consider programming education unnecessary and exhibit quite a high level of negative attitudes. In that case, the frequent negativity in the attitudes of both test and control group students might be the reason why there was no significant differentiation. Indeed, in the relevant literature, it is reported that among the greatest barriers before programming education are negative perception, motivation and attitude and these factors are much greater factors compared to the rest [27, 28, 33]. In the related studies, it is reported that in computer programming education students not only experience lower level of motivation but also some problems in cognitive domain as well [34-36].

On the other hand, education via the Scratch-based game activities contributes significantly to the students' academic achievements in C++ programming language unlike the traditional method. In the Scratch environment, the programming structures are more physical (visual) and the findings can be monitored physically in a thematic environment which might be the explanation of the positive contribution to the students' algorithmic

thinking skills. According to Bassey, Afuro, and Munienge," [37] students learned programming via the applied development this finding is consistent with the relevant literature. Relevant studies manifest that Scratch can contribute to students' performance in a variety of manners. For instance, Genç and Karakuş [23] in their study put forth that students are mostly endowed with positive views on Scratch and learning via designing guarantees permanent learning and they adopt a blog-supported education method. In Kobsiripat's [38] study, which was conducted to designate the effects of programming via Scratch on the creativity level, it was emphasized that this environment can be utilized in learning activities and improve students' creativity skills. Likewise, in the studies conducted by Kordaki [39], Ferrer-Mico at. Al. [40] and Garcia Quan [41] similar findings were detected.

It can be said that there is not enough research in literature related to the efficiency of the new method, approach or strategies aiming to solve problems faced in learning programming. Although some research can be found related to the usage of Scratch at the different levels of programming education, there are very few studies about its usage at the graduate level, especially at the faculties of engineering. In this regard, this research was designed to determine the usefulness of Scratch for programming education at the graduate level and it was concluded that it especially contributes to the academic success of students. In the light of these findings, it can be suggested that the Scratch-based educational applications can be employed at the onset of a computer programming course to develop algorithmic thinking and programming skills of undergraduate students who have just started to take programming education.

## ACKNOWLEDGEMENT

Some part of this research was submitted as an oral presentation in Erpa 2015.

This research has been executed within the scope of SEB-BAP 14-37 project endorsed by Amasya University Project Coordination Unit of Scientific Researches

## REFERENCES

[1] Ö. Korkmaz, H. Altun.. A Validity and reliability study of the Learning Computer Programming Attitude Scale (LeCoPAS). *Mevlana International Journal of Education*, 2014a, 4(1): 30-43 http://dx.doi.org/10.13054/mije.13.73. 4.1.

[2] X. Fang. Application of the participatory method to the computer fundamentals course, Affective Computing and Intelligent Interaction. *Advances in Intelligent and Soft Computing*, 2012, 137: 185-189.

[3] Ö. Korkmaz.. The Impact of Critical Thinking and Logical-Mathematical Intelligence on Algorithmic Design Skills. *Journal of Educational Computing Research*, 2012, 46(2):173-193. DOI: 10.2190/EC.46.2.d.

[4] W.W.F. Lau, A.H.K. Yuen. Modelling programming performance: Beyond the influence of learner characteristics. *Computers & Education*, 2011, 57: 1202–1213. doi:10.1016/j.compedu.2011.01.002

[5] Y. Wang, H. Li, Y. Feng, Y. Jiang, and Y. Liu. Assessment of programming language learning based on peer code review model: Implementation and experience report. *Computers & Education*, 2012, 59: 412–422. doi:10.1016/j.compedu.2012.01.007.

[6] R. Hamada. *The Relationship between Learning Logo and Proficiency in Mathematic*s. Unpublished Doctorate Thesis. Colombia: Colombia University, 1986.

[7] E. Çetin. *The effect of computer programming education on children's problem-solving skills*. Unpublished Master Thesis. Gazi Universities, Ankara, 2012.

[8] Y. Akpınar and A. Altun, Bilgi ToplumuOkullarında Programlama Eğitimi Gereksinimi [Programming Educational Needs in the Information Society School]. *Elementary Education Online*, 13(1), pp: 1-4.

[9] A. Gomes, A. J. Mendes. Learning to program-difficulties and solutions. *International Conference on Engineering Education, ICEE*. 2007.

[10] P. Tan, C. Ting, S. Ling. Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception. *International Conference on Computer Technology and Development*, 2009, 42-46.doi:10.1109/ICCTD.2009.188.

[11] T. Jenkins. On the difficulty of learning to program. *in Proc. of the 3 rd Annu. LTSN_ICS Conf*., Loughborough University, United Kingdom, 2002, pp. 53-58.

[12] Z. Katai, K..Juhasz, and A.K. Adorjani. On the role of senses in education. *Computers & Education*, 2008, 51(4), 1707–1717.doi:10.1016/j.compedu.2008.05.002.

[13] Ö. Korkmaz. Students' Difficulties in and Opinions about Designing Algorithms According to Different Instructional Applications. *Energy Education Science and Technology Part B: Social and Educational Studies*, 2013, 5(1).209-218.

[14] I. Milne, and G. Rowe. Difficulties in Learning and Teaching Programming—Views of Students and Tutors. *Education and Information Technologies*, 2002, 7(1): 55-66.

[15] J.P. Landry, J.H. Pardue, M.V. Doran, and R.J. Daigle. Encouraging Students to Adopt Software Engineering Methodologies: The Influence of Structured Group Labs on Beliefs and Attitudes. *Journal of Engineering Education*, 2002, 91(1):103-108.

[16] M., Resnick, Y. Kafai, J. Maloney, N. Rusk, L. Burd, and B. Silverman. A Networked, Media-Rich Programming Envi-ronment to Enhance Technological Fluency at After-School Centers in Economically Disadvantaged Communities. *Proposal to National Science Foundation*, 2003.

[17] M. Yorulmaz. Internet Kafelerin Daha FaydalıKullanılabilmeleri için Bir Öneri: Scratch [Recommendations for a More Useful can use the Internet Café: Scrathc]. *inet-tr' 2008 :22-23 December*. METU: Ankara.

[18] N. Çağıltay Ercil, and M. Fal. *Scratch İle Programlamayı Öğreniyorum [I learn programming with Scratch]*. Ankara: METU Pub, 2013.

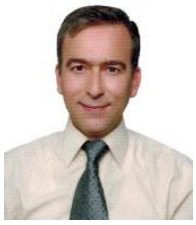
[19] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. The Scratch Programming Language and Environment. *ACM Transactions on Computing Education*, 2010, 10(4), 1-16.

[20] B. Kaučič, and T. Asič. Improving Introductory Programming with Stratch? *MIPRO*, 2011, p:1095-1100.

[21] J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, and M. Resnick.. Scratch: A Sneak Preview. *Second International Conference on Creating, Connecting, and Collaborating through Computing*. Kyoto, Japan, 2004, pp. 104-109.

[22] D. Karabak,, and A.Güneş. Curriculum Proposal for First Class Secondary School Students in the Field of Software Development. *Journal of Research in Education and Teaching*, 2013, 2(3):175-181.
[23] Z. Genç, S. Karakuş. Learning Through Design: Using Scratch In Instructional Computer Games, Design. *5 th International Computer & Instructional Technologies Symposium*, 22-24 September. Fırat University, Elazığ, 2011, p: 981-987.
[24] N. Calder. Using Scratch: An Integrated Problem-solving Approach to Mathematical Thinking. *APMC*, 2010, 15 (4), 9-14.
[25] G.Hwang, P. Wu, and C. Chen.. An online game approach for improving students' learning performance in web-based problem-solving activities. *Computers & Education*, 2012, 59: 1246–1256. doi:10.1016/j.compedu.2012.05.009.
[26] C. Lai, Q. Wang, and J. Lei. What factors predict undergraduate students' use of technology for learning? A case from Hong Kong. *Computers & Education*, 2012, 59(2): 569–579.doi:10.1016/j.compedu.2012.03.006.
[27] S.D. Anastasiadou, and A.S. Karakos. The beliefs of electrical and computer engineering students' regarding computer programming. *The International Journal of Technology, Knowledge and Society*, 2014, 7(1): 37-51.
[28] Y. Erdogan, E. Aydin, and Y.T. Kabaca. Exploring the Psychological Predictors of Programming Achievement. *Journal of Instructional Psychology*, 2008, 35(3): 264-270.
[29] H. Kotaman.. Self-Efficay Belif and Enhancement of Learning Performance. *Uludağ University the Journal of Educational Faculty*, 2008, XXI (1), 111-133.
[30] V. Ramalingam, and S. Wiedenbeck. 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*, 1998, 19(4), 367-381.
[31] Ö. Korkmaz. H. Altun. Adapting Computer Programming Self-Efficacy Scale and Engineering Students' Self-Efficacy Perceptions. *Participatory Educational Research (PER*). 2014b. Vol. 1(1), pp. 20-31.
[32] B.G. Tabachnick, and L.S. Fidell. *Using Multivariate Statistics* (sixth ed.). Pearson, Boston, 2013.
[33] C. Sacks, Y. Bellisimo, and J. Mergendoller. Attitudes toward computers and computer use: the issue of gender. *Journal of Research on Computing in Educatio*n, 1993, 26: 257-269.
[34] N. Hawi. Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. *Computers & Education*, 2010, 54: 1127–1136.doi:10.1016/j.compedu.2009.10.020
[35] B.P. Hernane, F.Z. Gilney and A. Marcelo. Learning computer programming: Implementing a fractal in a Turing machine, *Computers & Education*. 2010, 55(2):767-776. doi: 10.1016/j.compedu.2010.03.009.
[36] A. Robins, J. Rountree, and N. Rountree. Learning and Teaching Programming: A Review and Discussion, *Computer Science Education*, 2003, 13(2): 137–172.
[37] I. Bassey, D.Afuro, and M. Munienge,"An Investigation of Software Engineering Knowledge of Undergraduate Students", *IJMECS*, vol.7, no.12, pp.42-50, 2015.DOI: 10.5815/ijmecs.2015.12.06.
[38] W. Kobsiripat. Effects of the Media to Promote the Scratch Programming Capabilities Creativity of Elementary School Students. *Procedia - Social and Behavioral Sciences*. 2015, 174: 227–232.doi:10.1016/j.sbspro.2015.01.651.
[39] M. Kordaki. Diverse Categories of Programming Learning Activities could be performed within Scratch. *Procedia - Social and Behavioral Sciences*, 2012, 46: 1162-1166.doi:10.1016/j.sbspro.2012.05.267.
[40] T. Ferrer-Mico, M. À. Prats-Fernàndez, A., Redo-Sanchezb. Impact of Scratch Programming on Students' Understanding of Their Own Learning Process. *Procedia - Social and Behavioral Sciences*. 2012, 46: 1219–1223. doi:10.1016/j.sbspro.2012.05.278.
[41] C. Garcia Quan. Student Teachers Evaluating and Assessing SCRATCH in the Applied Linguistics, 2015. Classroom. *Procedia - Social and Behavioral Sciences*. 174: 1450–1456.doi:10.1016/j.sbspro.2015.01.774.

## Authors' Profile



**Özgen KORKMAZ** was born in 1972 in Konya. He graduated from the Faculty of Industrial Arts Education, and started teaching in the Department of Computing in 1993 at Gazi Uni. He completed a MA degree in 1996 at the Institute of Sciences Department of Computer Education at Afyon Kocatepe University. In 2007, he completed a PhD at the Institute of Educational Sciences, Department of Educational Technology at Gazi University. He is currently working as an Associate Professor at Amasya University at the Faculty of Technology Department of Computer Engineering. His research interests include computer programming, web-based learning, blended learning, message design and web-based programming technologies.