Participatory Educational Research (PER)
Special Issue 2015-II, pp., 27-34; 5-7 November, 2015
Available online at http://www.partedres.com
ISSN: 2148-6123
http://dx.doi.org/10.17275/per.15.spi.2.4

Learning Difficulties and Use of Visual Technologies in Learning to Program

Mehmet Fatih YİĞİT * and Mustafa BAŞER
Ondokuz Mayıs University, Faculty of Education, Computer Education and Instructional Technology Department, Samsun

Abstract
In recent years, programming has been seen as a promising career with great earning potential. Rapid growth and use of computer technology give further emphasis to programming. Therefore, faculties open programming courses ranging from introductory to advanced levels. Primary and secondary schools also started to include programming courses in their curriculum.
However, in the literature, it has been widely accepted that learning to program is a challenging task for students. Studies conducted on programming education show that significant number of students fail the programming course or get low grades. Of the reasons why students have problems in programming, necessity to possess variety of knowledge while dealing with the programming exercises is mentioned most in the research. These are syntactic, conceptual and strategic knowledge. Other reasons involve lack of motivation, lack of skills required for the programming, lack of mathematical knowledge and unfamiliarity with the programming courses.
This is a literature review study that discusses learning difficulties of students and visual technologies used in programming courses to overcome problems mentioned above. This study examines technologies such as Scratch, Alice, Blockly, Jeliot and Ville. The properties of these technologies, how they can be used in programming and results of research studying the effectiveness of these technologies are also discussed

Keywords: difficulties in programming, visual technologies in programming, Scratch, Alice, Blockly, Jeliot, Ville.

Introduction

Programming is a basic skill that students in Computer Science field are required to learn. However, it is also one of the subjects in educational institutions that most students find difficult to learn. Majority of studies conducted on programming education mention this problem. These studies also examine the causes and effects of this problem and recommend ways to deal with it (e.g., Eckerdal, McCartney, Moström, Ratcliffe & Zander, 2006; Lahtinen, Ala-Mutka & Jarvinen, 2005; Lister et al., 2004; McCracken et al., 2001; Robins, Rountree & Rountree, 2003).

* mehmetfatih.yigit@omu.edu.tr

Learning Difficulties in Programming

There are many programming difficulties cited in the literature. Some of them are:

- Programming requires more than one type of knowledge (Cooper, Dann and Pausch (2000a; Mannila, Peltomäki and Salakoski, 2006). These are syntactic, conceptual and strategic knowledge (Bayman & Mayer, 1988; Linn, 1985).
- Students are unfamiliar with programming courses (Jenkins, 2002) unlike other courses such as math or science that they met almost at the beginning of their school life.
- Lack of motivation to study programming is also another factor that makes learning to program difficult (Gomes and Mendes, 2007).
- Lack of mathematical knowledge also contributes to difficulty of programming (Gomes, Bigotto and Mendes, 2006).

Types of programming languages are seen as an effective factor playing important role in student success in programming (Goldwasser & Letscher, 2008). Object oriented programming requires students to learn complicated subjects like classes or objects in addition to some basic concepts such as variables, functions and loops, which makes programming more difficult and increases total amount of mental work of students (Cooper, Dann & Pausch, 2003; Proulx, Raab & Rasala, 2002).

Visual Technologies in Programming

It has been widely discussed in the literature that learning to program is a difficult process for students. However, there are attempts to overcome this problem and the most common method is use of visual technologies in programming education. There are several underlying reasons about integration of such technologies into programming classes. Kaučič and Asič (2011) stated that studies that examine the effects of these technologies in programming education obtained similar results, which are positive student attitudes towards programming courses and positive effects on student motivation. Studies also showed that using visual techologies within programming classes concretize the abstraction of programming and helps understanding and remembering of programming concepts (Naps et al., 2002; Shu, 1999).

Learning styles of students can become an important factor in learning to program. Several research taking both programming and learning style together into consideration revealed that most students have visual learning styles (Allert, 2004; Chen & Lin, 2011; Gomes & Mendes, 2008; Gomes & Mendes, 2010; Kuri & Truzzi, 2002; Thomas, Ratcliffe, Woodbury & Jarman, 2002). Therefore, considering the learning styles of students, use of visual technologies in programming education may turn out to be an effective option. Visual technologies used in programming education are divided into two categories as visual programming languages and program visualization tools (Myers, 1990).

There are some Visual Programming Languages (or tools) available for teaching and learning programming. These tools designed to overcome learning difficulties of students. One of the is Scratch. Scratch is a programming software in which users drag and drop visual blocks to build a program (Meerbaum-Salant, Armoni & Ben-Ari, 2011). Scratch enables users to meet programming concepts and develop their problem solving skills prior to programming syntax

(Malan & Leitner, 2007). As seen in *Figure 1*, Scratch has four main areas. These are block area, where programming blocks are located, script area, where users drag and drop blocks to create a program, stage area, where program output is shown visually and character area, where user select a character to place on the stage. Users first choose characters and then drag and drop blocks onto the program building area. According to the program output, some changes occur about the characters and the users can observe them in the stage area. As the programs are created with blocks, it is impossible for users to make errors related to syntax.

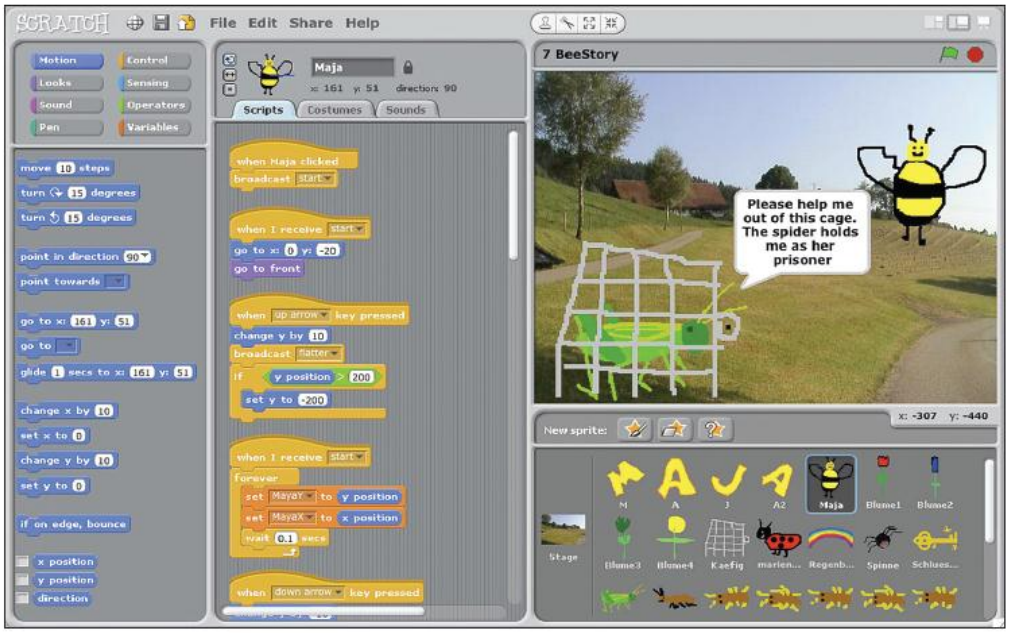

Figure 1: Scratch user interface

Considering these features of Scratch, Kordaki (2012) stated that Scratch has an important contributions to programming learning process such as providing an interactive learning environment, emphasizing programming concepts, saving users from cognitive load, supporting problem solving, visualizing program outputs, providing immediate feedback and increasing users' motivation. Findings of some research in which Scratch was used as learning tool can be summarized as:

- If students were taught some programming skills with Scratch prior to programming course 74% of them passed. On the other hand only 39% of the freshman programming students passed if they hadn't received any Scratch (Rizvi & Humphries, 2012).
- According to the findings of research done by Malan and Leitner (2007), 76% of the students received Scratch based course developed positive attitude toward programming.
- Ozorona, Çağıltay and Topallia (2012) found that Scratch supported programming course makes the learning environment funnier and more attractive to students, makes it easier to learn algorithm, improves student creativity, decreases the fail rate of the course and increases student participation.

Taking into consideration the features of Scratch and results of studies about Scratch, it is

possible mention that Scratch is a useful visual programming software for new learners. Scratch can visualize and concretize the difficult and abstract parts of programming and introduce the programming concepts to beginning learners in a way that they can understand better.

Alice may be another choice for learners who have difficulties with learning programming. Alice seems to help students to learn programming (Brown, 2008; Cooper, Dann & Pausch, 2000b; Sykes, 2007), increases students' self confidence in programming and makes students develop positive attitudes towards programming (Courte, Howard and Bishop-Clark, 2006), improves students achievement in programming (Sykes, 2007), and makes learning to program easier (Wang, Mei, Lin, Chiu & Lin, 2009).

Blockly is also a famous visual programming tool developed by Google (Blockly, 2015). Similar to Scratch and Alice, in Blockly users build their program by dragging and dropping visual blocks. The main feature that may separate Blockly from other VPL is that Blockly is able to show the corresponding syntactical codes of program created by blocks in JavaScript, Python, PHP and Dart programming languages. In other words, users can see the actual written code of the program they build with blocks. This distinctive feature of Blockly enables users to migrate to conventional text-based languages. It can be mentioned that Blockly serves more realistic learning environment than Scratch or Alice as it does not involve in relatively unnecessary things for programming such as designing characters or adding sounds and pictures to the stage. In Blockly, users only concentrates on program building exercises by using blocks which can support important programming subjects such as variables, loops, functions, conditionals and lists.

Program visualization tools (PVT) are another type of visual technology used in programming education in order to overcome learning difficulties of students. VPL helps students during program building process via graphical elements. On the other hand, PVT helps them after the program is executed. In other words, PVT shows the execution of program progresses by highlighting parts of the code under execution or by visualizing changes in variable states (Laakso, Kaila & Salakoski, 2008). By the help of this feature, students are able to see what happens while the program is under execution and what changes occur at different steps of the program.

The most widely used program visualization tool is Jeliot. Jeliot make visualization in Java programming language. As seen in *Figure 2*, Jeliot is consisted of three main parts. In the left panel, the actual written program codes are located. Users can write the code to this panel in Java. When they run the program, they can follow the program execution progress step by step in an animated way in the right panel of the software. The bottom panel shows the output of the program.

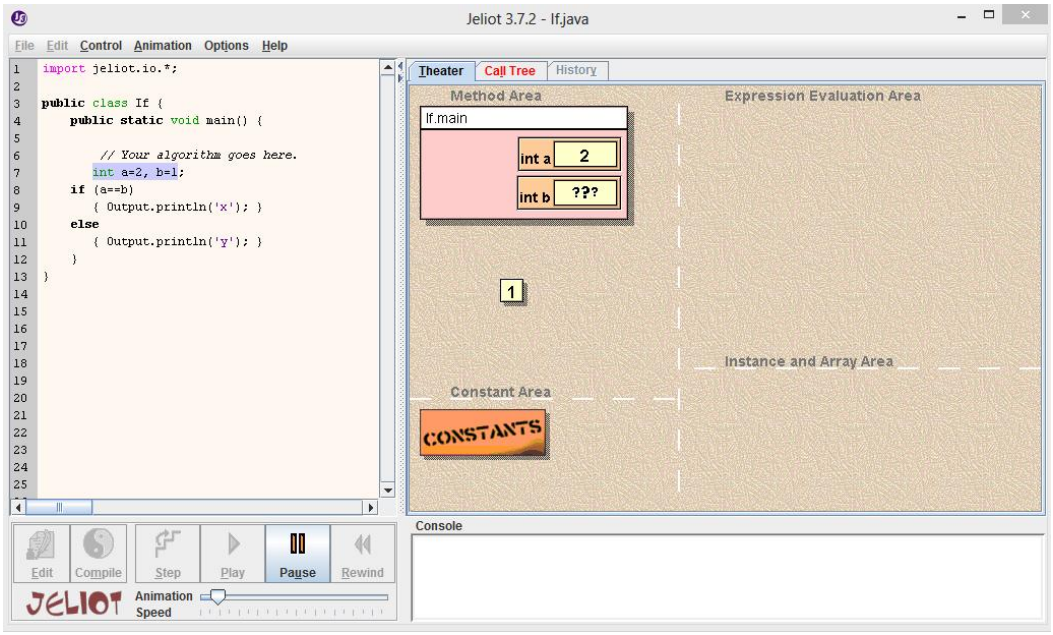

Figure 2 : Jeliot user interface.

The findings of studies conducted on the effects of Jeliot on student programming learning show that :

- Jeliot makes some programming concepts such as loops and conditionals more understandable to students and provides them with the oppurtunity to better see the changes in various states of the program (Kannusmäki, Moreno, Myller & Sutinen, 2004).
- Jeliot increases student achievement in the exercises in which they are supposed to predict program outputs and helps the average students by enabling them to build concrete models of how the program works inside (Ben-Bassat Levy, Ben-Ari & Uronen, 2004)

Discussion

In this study, students' learning difficulties of programming and visual technologies used to overcome these difficulties are discussed. The main difficulty arises from the fact that programming requires multiple types of knowledge at the same time, which are syntactic, conceptual and strategic knowledge. Other difficulties are related to student motivation, mathematical background, unfamiliarity to programming and languages used in programming. In order to get rid of this problem in programming education, the use of visual technologies are recommended in the literature. These are visual programming languages such as Scratch, Alice and Blockly and programming visualization tools such as Jeliot and Ville. The former helps students in the program building process with the help of visual blocks, while the latter helps them while the program is executed. The results of studies which examine the effects of these two types of technologies can be considered as positive. They play an important role in increasing student achievement and motivation and improving student attitudes towards programming courses. However, it may not be said that the number

of research about these technologies are adequate. For this reason, the effectiveness of these visual technologies on student learning to program should be further investigated.

References

Allert, J. (2004, August). Learning style and factors contributing to success in an introductory computer science course. In *Proceedings of the fourth IEEE international conference on advanced learning technologies (ICALT'04)* (pp. 385–389). Washington, DC: IEEE Computer Society.

Bayman, P. ve Mayer, R., (1988), Using Conceptual Models to Teach BASIC Computer Programming, *Journal of Educational Psychology*, *80*(3), 291-298.

Ben-Bassat Levy, R., Ben-Ari, M., & Uronen, P. A. (2003). The Jeliot 2000 program animation system. *Computers & Education*, *40*(1), 1-15.

Blockly (2015). https://blockly-demo.appspot.com/static/demos/code/index.html, retrieved on 11.10.2015.

Brown, P. H. (2008). Some field experience with Alice. *Journal of Computing Sciences in Colleges*, *24*(2), 213-219.

Chen, L., & Lin, J. M. C. (2011). Learning styles and student performance in java programming courses. In *Proceedings of the 2011 international conference on frontiers in education: CS and CE* (pp. 53-58).

Cooper, S., Dann, W., & Pausch, R. (2000a). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges*, *15(5)*, 107-116.

Cooper, S., Dann, W., & Pausch, R. (2000, November). Developing algorithmic thinking with Alice. In *Information Systems Educators Conference* (pp. 506-539).

Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. *ACM SIGCSE Bulletin*, *35*(1), 191-195.

Courte, J., Howard, E., & Bishop-Clark, C. (2006). Using Alice in a computer science survey course. *Information Systems Education Journal*, *4*, 1-7.

Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., & Zander, C. (2006). Can graduating students design software systems?. *ACM SIGCSE Bulletin*, *38*(1), 403-407.

Goldwasser, M. H.,& Letscher, D. (2008). Teaching an object-oriented CS1 with Python. *ACM SIGCSE Bulletin, 40(3),* 42-46.

Gomes, A., Carmo, L., Bigotte, E., & Mendes, A. (2006, September). Mathematics and programming problem solving. In *3rd E-Learning Conference–Computer Science Education* (pp. 1-5).

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

Gomes, A., & Mendes, A. (2008, June). A study on student's characteristics and programming learning. In *World Conference on Educational Multimedia, Hypermedia and Telecommunications* (pp. 2895-2904).

Gomes, A. J., & Mendes, A. J. (2010, June). A study on student performance in first year CS courses. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (pp. 113-117). ACM.

Jenkins, T. (2002, August). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences* (pp. 53-58).

Kannusmäki, O., Moreno, A., Myller, N., & Sutinen, E. (2004, July). What a novice wants: students using program visualization in distance programming course. In *Proceedings of the Third Program Visualization Workshop (PVW'04)* (pp. 126-133).

Kaučič, B., & Asič, T. (2011, May). Improving introductory programming with Scratch?. In *MIPRO, 2011 Proceedings of the 34th International Convention* (pp. 1095-1100). IEEE.

Kordaki, M. (2012). Diverse categories of programming learning activities could be performed within Scratch. *Procedia-Social and Behavioral Sciences*, *46*, 1162-1166.

Kuri, N. P.,& Truzzi, O. M. S. (2002). Learning styles of freshmen engineering students. In *Proceedings, 2002 International Conference on Engineering Education*.

Laakso, T. R. M. J., Kaila, E., & Salakoski, T. (2008). Effectiveness of program visualization: A case study with the ViLLE tool. *Journal of Information Technology Education*, *7*, 15-32.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. (2005). A study of the difficulties of novice programmers. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on innovationand technology in computer science education, Caparica* (pp. 14–18). Portugal: ACM Press.

Linn, M. C. (1985). The cognitive consequences of programming instruction in classrooms. *Educational Researcher*, *14*(5), 14-29.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., ... & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, *36*(4), 119-150.

Malan, D. J.,& Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM SIGCSE Bulletin*, *39(1)*, 223-227.

Mannila, L., Peltomäki, M., & Salakoski, T. (2006). What about a simple language? Analyzing the difficulties in learning to program. *Computer Science Education*, *16*(3), 211-227.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B. D., ... & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, *33*(4), 125-180.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education*, *23*(3), 239-264.

Myers, B. A. (1990). Taxonomies of visual programming and program visualization. *Journal of Visual Languages & Computing*, *1*(1), 97-123.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., ... & Velázquez-Iturbide, J. Á. (2002). Exploring the role of visualization and engagement in computer science education. *ACM SIGCSE Bulletin*, *35(2)*, 131-152.

Ozoran, D., Cagiltay, N., & Topalli, D. (2012). Using Scratch In Introduction to Programming Course for Engineering Students. In *2nd International Engineering Education Conference (IEEC2012)* (pp. 125-132).

Proulx, V. K., Raab, J., & Rasala, R. (2002). Objects from the beginning-with GUIs. *ACM SIGCSE Bulletin*, *34*(3), 65-69.

Rizvi, M., & Humphries, T. (2012, October). A Scratch-based CS0 course for at-risk computer science majors. In *Frontiers in Education Conference (FIE), 2012* (pp. 1-5). IEEE.

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

Shu, N. C. (1999). Visual programming: Perspectives and approaches. *IBM Systems Journal*, *38*(4), 199-221.

Sykes, E. R. (2007). Determining the effectiveness of the 3D Alice programming environment at the computer science I level. *Journal of Educational Computing Research*, *36*(2), 223-244.

Thomas, L., Ratcliffe, M., Woodbury, J., & Jarman, E. (2002). Learning styles and performance in the introductory programming sequence. *ACM SIGCSE Bulletin,34(1*), 33-37.

Wang, T. C., Mei, W. H., Lin, S. L., Chiu, S. K., & Lin, J. M. C. (2009, October). Teaching programming concepts to high school students with alice. In *Frontiers in Education Conference, 2009. FIE'09. 39th IEEE* (pp. 1-6). IEEE.