Choice of Visual Programming Language for Learning Programming

OLIVERA ISKRENOVIC-MOMCILOVIC
University of Novi Sad, Faculty of Education
Podgoricka 4, 25000 Sombor
SERBIA
oljkaisk@yahoo.com

Abstract: - Programmers beginners learning programming equate with learning the syntax of a programming language, but fail to master skills such as algorithmic thinking and problem solving. Problems in learning programming occur in novice programmers at all ages. Today there are many visual programming languages, which can help developers to beginners in a simple and interesting way to learn about the basic concepts of programming, and some of them even do not require learning a programming language syntax. The aim of this work is the choice of visual programming language, which makes it easier for beginners learning programming. Multi-criteria analysis has shown that the most suitable for learning Scratch programming.

Key-Words: - beginners, learning, programmers, programming, visualization, programming languages

1 Introduction

Today the challenge is learning how to program in the field of education. It enables the development of problem-solving skills at the beginner programmers. The problems, which are solved in learning programming can be from various fields such as mathematics, physics, chemistry .... However, the area of problem solving, which can create problems if the developers beginners do not realize the problem. On the other hand, when learning to program, developers beginners need to understand how their program works, or should create a link between what you wrote in your program and what the program is running, as it will otherwise have problems in detecting errors when the program does not work as they imagined [1].

Programming can be difficult for beginners, because they need to learn a specific syntax, which may be incomprehensible and confusing [2]. Beginners do not understand concepts such as variables, branching, loops, recursion, input and output commands, as well as how all these terms expressed by command programming language [3]. All this inevitably creates difficulties in preparing the algorithms and writing programs. On the other hand, it can be noticed that the students who have learned the syntax of the language and simple concepts often fail to assemble the correct program. This shows that beginners need a lot of work and time to understand the abstraction of certain programming languages.

Today they made a number of programming languages and software environments with the intention of programming becomes available to multiple users [4]. Young people who use computers usually are not programmers, but would like to create their own programs, which are similar to those programs that use everyday, such as interactive and graphically rich games, and educational simulations [2]. Setting the start of a programming language is not an easy problem, especially if we take into account the consequences of that choice may have on future work and success of the rookies [5]. Too complex programming language will rapidly cause aversion and demotivation for beginners. Therefore, the initial programming language had to have a simple syntax, fast feedback and structured way of writing beginners to acquire the habits of proper writing programs [6]. More and more lecturers claim that some people just can not learn programming and programming languages. Some authors argue that most beginner programmers at the beginning could acquire the basic knowledge and skills of programming in the context of computer literacy, without having to later become experts in programming [7]. Programming does not provide in itself a lot of knowledge and skills for beginners [8, 9], because it is present the abstract nature of the contents, you must understand and realize example [10].

Today it is considered that the complex syntax of programming languages too burdened by beginners, and thus developed the so-called. mini languages (mini languages) [11], that is, languages that are simplified and developed for learning. Such

languages are limited but allow for the transition to more complex programming languages. Logo programming language developed by Papert is considered one of the first such mini-languages, after it formed many others with a similar idea of visualization, so called visual programming languages (visual programming languages). Visualization is introduced as an aid to understanding asptraktnih content programming languages [12] starting from the visual display of the syntax of programming languages to visual display program execution. Beginners point out how they have always confusing syntax of the language and demanding [4]. In the beginning, learning the syntax of a programming language looks like learning a foreign language, but it is much more difficult because it is a programming language often abstract and incomprehensible. Aggravating circumstance is that beginners have to overcome an additional environment in which to create a program that reason problem, solve, to understand the art of solving problems and to find an optimal solution (caption program), which will eventually be tested.

Visual programming languages can be divided into:

1. pure visual programming languages,
2. hybrid visual programming languages.

For pure visual programming languages, the programmer creates a program of graphic elements, which are elements of a programming language (commands, variables, etc.) [13]. The program is executed in the same graphical environment. There is no translation or transition to text mode. Hybrid visual programming languages have a combination of graphical environment with textual form. Practice has shown that better and more skilful quick primer "outgrow" a visual form of commands and switch to text mode [13]. Scratch is an example of pure visual programming language, which yields opportunities for learning programming ideas to the Logo programming language. Apart from the aforementioned *Scratch*, often used other languages such as: *Alice, Kodu, eToys, App Inventor, Alice, Greenfoot, …*

The above visual programming languages are effective for initial learning programming, because they represent the graphic environment in which to play and create games [13]. In order to link the two worlds, programming and game, there is the idea of learning through game programming. For this form of learning programming are suitable visual programming languages. In this way beginners is fun and interesting to program. Thus, shifting the focus of learning is done with the syntax of the semantic structure of the program. The developers are not aware of beginners to practice troubleshooting, debug programs and to create a variety of programs, but they are convinced that creating computer games. Encouraging beginners to learn one until they think they are doing something completely different, Pauch called spoofing head (head fake) [14].

2 Research Methodology

This paper presents a multi-criteria analysis of visual programming languages. In recent decades a great number of methods of multi-criteria analysis. They are used daily in decision-making greater or less importance. The problem of multi-criteria analysis boils down to a comparison of alternatives evaluated according to a number of different criteria (usually of different relevant character) using an appropriate budget. For a given set of alternatives and provide a set of criteria, a natural question that arises is which alternative is best? For the evaluation and ranking of alternatives is used network method, which was introduced by Malisa Žižovic and Damljanovic Nada [15].

Using network analysis methods of multi-criteria are defined alternatives and criteria for the application of visual programming languages. A set of alternatives includes:

A1: *Logo* - programming through drawing the turtle giving orders, where she needs to go and when to start to draw, but when you move without drawing [16, 17]

A2: *Kodu* - programming of computer game and creating 3D virtual worlds through the understanding of the code whose elements are image icons [18]

A3: *Scratch* - programming by dragging and fitting visual components that are similar to Lego blocks and can be connected only if it corresponds to the specific terms of syntax [19]

A4: *Alice* - allows the creation of animations through reading the script, design, writing and testing programs [20]

Given a set of alternatives is evaluated based on four criteria:

C1: *System requirements and installation process* - processor, memory, graphics card, network card, operating system installation guide

C2: *Visual impact* - attractive appearance, readability of messages, clear visual representations, reality graphics, pleasant color combinations, interesting graphics

C3: *User interface and navigation* - complexity and clarity of command and control to work, stop work

at any time, separately-use, intuitive navigation, this additional help and documentation for work

C4: *Suitability for beginners* - the speed of learning, the complexity of the program, the possibility of making a syntax error, speed, fault-finding, the possibility of correction of errors

The assumption is that all criteria maximizing type (brought on the variant) and that the scale of 1 to 10 (with what is the best score of 10, and 1 weakest score). It is clear that it is possible to set and many other criteria when it comes to choosing a visual programming language, but these criteria can meet the needs of choice in most cases.

3 Research Results and Discussion

For the purposes of this study teachers, who teach informatics in primary schools and participate directly in the implementation of educational activities are evaluated visual programming language for learning programming. Expert evaluation of these alternatives to the proposed criteria are shown in Table 1.

Table 1. *Expert evaluation of alternatives according to the criteria*

	C1	**C2**	**C3**	**C4**
A1	6	6	7	8
A2	4	6	9	7
A3	7	7	10	8
A4	9	8	8	7

All teachers agree that in terms of system requirements and installation process is the best Alce, then *Scratch, Logo* and eventually *Kodu*. *Kodu* is a visually oriented than listed four visual programming language, when it is the most emphasis on graphics, but requires more memory and better graphics card [18]. The installation process of *Scratch* and *Alice* is simple, a little more complex with the *Logo*, while *Kodu* requires the previous installation of additional programs [21]. For all these visual programming languages on their official website, there are additional instructions, which can be used to help install new software or as an additional document found on the installation CD.

The best visual impact on developers beginners leaves *Alice, Scratch*, and then at the end of *Kodu* and *Logo*. The use of color and graphics when *Alice* is appropriate for beginners programmers, so that its appearance on the most motivated to work and learning [22]. The graphics, animation, music and video effects allow beginners to express themselves creatively to develop logical thinking and better understand the programming language.

Estimates for the criteria user interface and navigation are formed so that the score of 10 for "best" alternative data *Scratch*, and remained by 1 row less, because there are very small differences among these visual programming languages. *Scratch* has a very simple user interface, as the command and control of the work made clear [23]. Navigation is designed to enable intuitive orientation [24].

All experts agree that *Logo* and *Scratch* well suited for beginners programmers, then *Code* and *Alice* [25]. After acepting basic rules of the programming language and the possibility environment, developers beginners solve various problems creating appropriate program. This may be different animated stories, TV commercials, presentations lessons, computer games and the like. Programming is by *Logo* and *Scratch c*onceived so that novices can not make syntax errors [26]. If beginners make logical errors, they can be easily found and corrected [19].

Multi-criteria analysis of alternatives proposed under given criteria is made on the basis of network methods for multi-criteria evaluation [15]. This method is based on calculating the distances date from alternative A hypothetical ideal - the best alternative A * i hypothetical worst alternative *A. The bill includes the whole range of parameters, ranging from those enforcing the usual take into account the importance of the decision, to those who are for this specific methods such as functions that describe the significance of the differences between the alternatives. To say that one alternative is better than another if it is closer to the hypothetical best alternative A distant and hypothetical worst alternative *A. If the decision maker wants to have an alternative to the order, then we say that we should carry out balancing within hypothetical best alternative A* i hypothetical worst alternative *A. For this purpose, define the distances appropriate alternative A of the hypothetical best alternative A* i hypothetical worst alternative A* as

$$D(A) = D0(A) – D1(A).$$

Table 2. *Distance of appropriate alternative hypothetical best and worst alternatives*

	A1	**A2**	**A3**	**A4**
distance	0.175	0.0625	0.325	0.1667
alternatives	0.216	0.4735	0.175	0.25
from ideal	-0.041	0.3750	0.15	-0.0833

The final order of the alternatives:

$$A3 \rightarrow A1 \rightarrow A4 \rightarrow A2 ,$$

thus highlighted the advantage of alternative *A3* compared to other alternatives compared,. Signs on all the criteria for developers beginners the best visual prograsmski language *Scratch, Logo then, Alice* and *Kodu.*

4 Conclusion

Learning programming is difficult and demanding, and is the subject of numerous scientific and research papers. Problems with programming are present at all levels of education, especially problematic for beginners who first encounter with the abstract content of the programming language, and it must be adopted in a relatively short time.

Learning programming is often reduced to learning the syntax, but is ignored algorithmic problem solving, which is often problematic. Developers and beginners have problems in monitoring the finished program. Sometimes learning programming with the help of visual programming languages becomes too easy for beginners better or weaker beginners are not able to follow the simple programs. Some beginners recognized gives the reason for their failure is not one hundred practicing enough.

The choice of visual programming language is an important factor in learning programming. In this paper, using multi-criteria analysis has shown that Scratch good in every way for developers to beginners. Removing the complexity of the syntax allow beginners to the fun environment easier to adopt the necessary skills to solve complex problems fast. Reactions are generally positive beginners as quickly overcome learning programming.

References:

[1] W. Dann, S. Cooper, R. Pausch, *Learning to program with Alice*, Prentice Hall, New Jersey, USA, 2008.

[2] I. Durdevic, Procjene studenata uciteljskog studija o tri racunalna programa namijenjena malim pocetnicima u programiranju, *Radovi Zavoda za znanstveni i umjetnicki rad u Pozegi*, Vol.3, 2014, pp. 93-108

[3] M. Thune, A. Eckerdal, Variation theory applied to students' conceptions of computer programming, *European journal of engineering education*, Vol.34, 2009, pp. 339-347.

[4] C. Kelleher, Barriers to programming engagement, *Advances in gender and education*, Vol.1, No.1, 2009, pp. 5-10.

[5] G. Zaharija, S. Mladenović, I. Boljat, Introducing basic programming concepts to elementary school children, *Procedia - social and behavioral sciences*, Vol.106, 2013, pp.1576-1584.

[6] L. Mannila, Novices' progress in introductory programming courses, *Informatics in education*, Vol.6, No.1, 2007, pp. 139–152.

[7] J. Bennedsen, M. E. Caspersen, Persistence of elementary programming skills, *Computer Science Education*, Vol.22, No.2, 2007. pp. 81-107.

[8] M. Koling, The Greenfoot programming environment, *ACM Transactions on computing education*, Vol.10, No 4, 2010, pp. 182-196.

[9] M. Guzdial, Programming environments for novices, in Fincher, S. and Petre, M. (Eds.) *Computer science education research*, CRC press, Boca Raton, USA, 2004, pp. 127–154.

[10] E. Lahtinen, K. Ala-Mutka, H.M. Jarvinen, A study of the difficulties of novice programmers, *ACM SIGCSE Bulletin*, Vol.37, No.3, 2005, pp. 14–18.

[11] P. Brusilovsky, E. Calabrese, J. Hvorecky, A. Kouchnirenko, P. Miller, Mini-languages: a way to learn programming principles, *Education and information technologies*, Vol.2, No.1, 1997, pp. 65–83.

[12] M. Olsson, P. Mozelius, J. Collin, Visualisation and gamification of e-learning and programming education, *Electronic journal of e-learning*, Vol.13, No.6., 2015, pp. 441-454.

[13] M. Mladenovic, M. Rosic, S. Mladenovic, Comparing elementary students' programming success based on programming environment, *International journal of modern education and computer science*, Vol.8, 2016, pp. 1-10\

[14] R. Pausch, J. Zaslow, *The Last Lecture*, Hyperion, New York, 2008.

[15] M. Zizovic, N. Damljanovic, NNew method for multticriteria analysis, *UPB Scientific Billetin, Series A: Applied mathematics and physics*, Vol.73, No.2, 2011, pp. 13-22.

[16] W. Wang, *Beginning Programing all-in-one desk reference for dummies*, Wiley Publishing, Indiana, USA, 2008.

[17] B. Ward, D. Marghitu, T. Bell, L. Lambert, Teaching computer science concepts in Scratch and Alice, *Journal of computing sciences in colleges*, Vol.26, No.2, 2010, pp. 173-180.

[18] A. Fowler, T. Fristce, M. MacLauren, Kodu Game Lab: a programming environment, *Computer games journal*, Vol.1, No.1, 2012, pp. 17-28.
[19] J. Maloney, M. Resnick, N. Rusk, B. Silverman, E. Eastmond, The Scratch programming language and environment, *ACM Transactions on computing education*, Vol.10, No.4, 2010, article 16.
[20] I. Utting, S. Cooper, M. Kolling, J. Maloney, M. Resnick, Alice, Greenfoot, and Scratch – A discussion, *ACM Transactions on computing education*, Vol.10, No.4, 2010, Article 17.
[21] D. L. Kwong, M. Niibori, S. Okamoto, M. Kamada,T. Yonekura, Islay3D A programming environment for authoring interactive 3D animations in terms of state-transition diagram, *Journal of software engineering and applications*, Vol. =7, 2014, pp. 177-186.
[22] A. Ebrahimi, S. Geranzeli, T. Shokouhi , E. R. Tee, E.R. (2013). Programming for children: Alice and Scratch Analysis, *International Journal of information technology & computer science*, vol. 12, no. 3, 2013, pp. 106-115.
[23] F. Kaleliogluf, Y. Gulbahar, The effects of teaching programming via scratch on problem solving skills: A discussion from learners' perspective, *Informatics in education*, Vol.13, No.1, 2014, pp. 33–50.
[24] K. Asad, M. Tibi, J. Raiyn, Primary school pupils' attitudes toward learning programming through visual interactive environments, *World journal of education*, Vol.6, No.5, 2016, pp. 20-26.
[25] A. Sattar, T. Lorenzen, Teach Alice programming to non-majors, CM SIGCSE Bulletin, Vol.41, No.2, 2009, pp. 118-121.
[26] A. Y. S. Su, C. S. J. Huang, S. J. H. Yang, T. J. Ding, Y. Z. Hsieh, Effects of annotations and homework on learning achievement: An empirical study of scratch programming Pedagogy, *Educational technology & society*, Vol.18, No.4, 2015, pp. 331– 343.