Learning programming from Scratch

Conference Paper · July 2017

CITATIONS
0

READS
5,644

3 authors, including:

Monika Mladenovic
University of Split
14 PUBLICATIONS **43** CITATIONS
SEE PROFILE

Sasa Mladenovic
University of Split
38 PUBLICATIONS **85** CITATIONS
SEE PROFILE

LEARNING PROGRAMMING FROM SCRATCH

Monika Mladenović[1], Divna Krpan[1], Saša Mladenović[1]
[1] University of Split, Faculty of Science
Croatia
monika.mladenovic@pmfst.hr, divna.krpan@pmfst.hr, sasa.mladenovic@pmfst.hr

ABSTRACT
The link between problem-solving and programming skills is well known. Students with higher problem-solving abilities find programming easy and they can master programming with no or little difficulties regardless of the programming environment. On the contrary, students with lower problem-solving abilities find programming difficult to understand and are often unable to master it. The before mentioned groups of students usually make up two thirds of the entire class, the top and lowest thirds, respectively. What about the "middle third" students? This is probably the most represented group; those are students who can master programming but with some difficulties. Visual programming language environments are tools designed to engage all students but maybe the "middle third" students would gain the most benefit from that approach.
In this paper, we explore the educational and motivational effect of using Scratch for game-based programming on 5th-grade elementary school students based on their problem-solving abilities.
Results presented here confirm the positive effect of using Scratch as the introductory programming language for game-based programming on "middle third" students, compared to Python as the text-based programming language.
Keywords: programming, Scratch, problem-solving, elementary school, "middle third" students

INTRODUCTION

Programming is difficult, and most children won't become programmers, so why should they learn to program? Programming, as a part of Computer Science, is also a part of everyday life, so learning programming as soon as possible should be the educational right of the 21st Century (Maloney, Peppler, B. Kafai, Resnick, & Rusk, 2008).
It is well-known that programming requires higher problem-solving abilities and that by programming problem-solving abilities can be practised. Students with higher problem-solving abilities can master programming with no or little difficulties, so motivation is crucial to this group of students. On the other hand, students with low problem-solving abilities are facing a lot of difficulties during learning programming, and are often unable to master it. We could facilitate their efforts by choosing an appropriate pedagogical approach. Maybe the most interesting group are "middle-third" students who can master programming with some difficulties (Armoni, Meerbaum-Salant, & Ben-Ari, 2015). Motivation and appropriate pedagogical approach can be crucial for these students. Choosing the proper programming environment for target age may be the key factor.
Textual programming language approach rely on "real" programming approach with languages like BASIC, Python, etc. Those languages require high problem-solving skills and precision in syntax, so many children perceive programming as difficult. Programming is most commonly taught through solving various math problems. The question is: do children find programming based on this approach repulsive?
Visual programming languages are syntax error free and more attractive to children. Scratch is visual, block-based, programming language appropriate for elementary school children (Resnick et al., 2009). Using programming languages like Scratch has the advantage of eliminating syntax problems which allows students to focus on the language semantics. Besides that, elementary school students have yet to reach the appropriate level of abstract thinking required to program, which makes learning programming more difficult. Learning programming by using a visual programming language can provide concrete to abstract experience (Dann & Cooper, 2009), and can thus be used as a medium for mediated transfer (Dann, Cosgrove, Slater, & Culyba, 2012) towards "real" programming. By teaching programming in Scratch, we can also shift the context of programming from solving math problems, which is the most commonly used approach in text-based programming, to programming games, storytelling, etc. The question is: do students learn programming concepts or is it just fun?
Our previous research (Mladenović, Krpan, & Mladenović, 2016) showed that 5th-grade students with higher problem-solving abilities achieved better results in Python as "real", text-based programming language. When it comes to Scratch that is not the case. Students with intermediate problem-solving abilities achieved better results in Scratch than in Python. This led us to new questions, can we affect the "middle third" students to achieve better results in programming? Can we motivate the "higher third" students to take programming class? Can we measure the motivational factor of these languages? In this paper, we give answers to these questions. This research was conducted in classroom settings. We analysed 5th-grade elementary school students' test results

during their enrollment in elective course Informatics in two schools by both approaches using Python as textual language and programming games in Scratch. Students didn't have any programming experience, and their problem-solving abilities were tested prior to the experiment. Based on the problem-solving test results, students were assigned to one of three groups. Students were learning different programming concepts in two programming languages and were tested for both approaches. Test results were analyzed and compared for each problem-solving group. The results of our research are presented in this paper.

BACKGROUND

Novice programmers find it very difficult to master programming. In order to solve a programming problem, we first need to break the problem into smaller, more manageable steps. This is the process of developing an algorithm. Students who need to focus more on syntax commonly ignore this phase. Flow charts and trace tables are often used to take the students through the steps of structured problem solving which leads to identifying sequence, selection, and repetition (Whitfield, Blakeway, Herterich, & Beaumont, 2007). Afterwards, novices need to program these steps using some programming language. A programming language is perceived as a major obstacle for novices (McCracken et al., 2001), especially when it comes to elementary school children because novice often focus on programming language syntax rather than developing an algorithm. Visual programming languages may help with this issue since they have simpler syntax, which allows novices to focus on developing an algorithm (Grover & Pea, 2013).

It has been reported that students with lower mathematical skills can learn problem-solving and programming if provided with appropriate materials and the use of less complex visual tools (e.g. Java Trainer) before moving to IDEs which are considered more complex (Whitfield et al., 2007). It is also reported that children weren't aware that they were programming, they argued that they were making games, stories, interactive presentations in Scratch (Maloney et al., 2008). This phenomenon Randy Pausch called “head fake” (Dann & Cooper, 2009) (Pausch & Zaslow, 2008).

In one of our previous research, we compared students’ success in LOGO, which is a text-based, and Scratch, which is a block-based programming language. Results showed that students’ success in Scratch was better, especially with regard to the concept of a nested loop. There were only a few students who were able to fully understand basic programming concepts while using Logo, but after a switch to Scratch that number increased. When it comes to motivation, Scratch is far more positively accepted than Logo (Mladenović, Rosić, & Mladenović, 2016).

Therefore, we can conclude that the “middle third” students can benefit from using a visual programming language. Similar conclusions were made in other studies. It was demonstrated that the use of the Jeliot program animation system primarily benefited “middle third” students (Ben-Bassat Levy, Ben-Ari, & Uronen, 2003). The same conclusion was reported in a study where Scratch was used as an introduction to C# (Armoni et al., 2015). Besides, students can master basic programming concepts more quickly by using a visual programming language (Armoni et al., 2015) (Price & Barnes, 2015).

However, there are studies that indicate some possible bad habits of programming in Scratch. It’s reported that during programming in Scratch middle-school students developed *bottom-up programming* and *extremely fine-grained programming bad* habits, although researchers were satisfied by motivation and developed technical skills of students by programming in Scratch (Meerbaum-Salant, Armoni, & Ben-Ari, 2011). These habits may have been developed as part of “natural learning” based on “scenario based learning” which fits with the idea of Scratch programming approach. These habits shouldn’t be concerning (Gordon, Marron, & Meerbaum-Salant, 2012), especially when Scratch is used as part of formal learning in a classroom setting in which teachers can guide students.

Researchers analysed a total of 100 projects, and two other bad habits were discovered. The first one refers to character naming, where most students didn't change default names like *Sprite1*, *Sprite2*. Conversely, it was also reported that user variables are named correctly, i.e., semantically meaningful. A possible explanation is that when creating a new character, the name is given automatically which is not the case when adding new variables. The second bad habit is duplicating code in the same project which indicates that abstraction and modularization were not taught (Moreno & Robles, 2014). Teachers who use visual programming languages in their classroom need to be aware of bad habits to minimize their occurrence.

METHODOLOGY

In this study, qualitative and quantitative methods were used.

Research design

This research was conducted with the purpose to compare the basic programming concepts understanding, concerning two programming languages: Python and Scratch. The target group were elementary school students

with no previous programming experience. In the Republic of Croatia Informatics is an elective course (Ministry of Science Education and Sports of the Republic of Croatia, 2005) from 5th to 8th grade, and programming is only one of several main topics in each grade. Therefore, 5th-grade students were appropriate for this research. Since programming is related to problem-solving skills and there is a positive correlation between math and programming (White & Sivitanides, 2003), the administered pre-test was designed to test student problem-solving skills. Students were first exposed to Python programming for four weeks with two hours per week which makes a total of eight hours. The lectures included selected programming concepts: variables, input, print, sequencing and conditionals. Student skills in Python programming were tested afterwards. Three weeks later (after winter school break), we introduced students to programming in Scratch. We have selected a game-based approach and students were required to program simple games. They were introduced to basic programming concepts like sequencing, conditional and iteration. The lectures were held for two hours per week for three weeks, and afterwards students' understanding of concepts in Scratch was tested. Additionally, they were given a questionnaire about their attitude towards programming after learning both Scratch and Python.

The participants of the experiment were students from two elementary schools in Split, in both of which the first author of this paper was the teacher. Prior to the experiment, the teacher had five years experience of teaching computer science in elementary schools and four years of experience in teaching computer science at the undergraduate level.

The research design is shown in Table1.

Table 1: Research design

Experiment				
week		**Topic**	**New terms and instructions**	**New concepts**
Pre- test				
1	**Python**	Algorithms: sequencing, conditional and iteration	Algorithm, sequencing, conditional and iteration	Introducing algorithm term, basic algorithms: sequencing, conditional and iteration with examples from real life. Introducing to Python programming language.
2		Variable, input and output	Variable, input, print, int	Basic Python instructions, variable term and integers with examples in Python.
3		input processing, output process phases of the computer program	Arithmetic operations (+,-,*, /)	Solving simple problems in Python program using input, processing including basic arithmetic operations and output.
4		Conditional	If else	Solving simple problems including branching algorithm in Python using if else.
Python test, questionnaire about programming and python				
Three-week Christmas holidays				
1	**Scratch**	Aquarium simulation program	forward, left, right, repeat	sprites, concurrency, loops
2		Chasing ghosts game	If, variables	conditionals
3		Simple ricochet game	communication by messaging, conditional loops, Coordination and Synchronisation	loops with conditionals
Scratch test, questionnaire about programming and programming languages				

The primary goal of this research is to find the differences in students' results between pre-tests and tests following the Python and Scratch lectures. The second goal is to examine the differences in attitudes towards programming and the programming language used. Based on that, we defined the next hypotheses:

- H1 – Students with higher problem-solving abilities will be more successful in Python programming than students with lower problem-solving abilities.
- H2 - Students with higher problem-solving abilities will be more successful in Scratch programming than students with lower problem-solving abilities.
- H3 – attitude towards programming will be more positive after Scratch than after Python.

Participants

The research sample consisted of 54 5th grade students from two schools during the school year 2014/2015. Since programming was taught for seven weeks, some students didn't attend all of the lectures or tests. Hence, the final number of participants is 50, which includes 34 boys and 16 girls. Students had no previous programming experience, which means that this is their first contact with programming. Non-probability, purposive sampling (Cohen, Manion, & Morrison, 2013) was used, because our goal was to target pupils with no previous programming experience in elementary school.

Assessment instruments

The data was collected in three phases. In the first phase, students were tested for problem-solving abilities before the programming lectures began. In the second phase, they were introduced to basic programming concepts in Python like variables, input, print, sequencing and conditionals. Students' knowledge (acquired concepts) was tested using Python assignments. They also filled a short questionnaire about the attitude toward programming. During the last, third phase, students were learning basic programming concepts like sequencing, conditionals and iteration while programming games in Scratch. Again, after the third phase, their acquired programming concepts were tested using Scratch assignments. They also filled a short questionnaire about the attitude toward programming and programming languages.

Data analysis

Results were analysed by qualitative and quantitative techniques that are used for triangulation purpose (Cohen et al., 2013) to increase the validity of the findings. Kolmogorov-Smirnov test is used to determine the normality of data. Parametric independent t-test and non-parametric Mann-Whitney U test are used to compare results between groups. Parametric test paired t-test and non-parametric Wilcoxon Signed Ranks Test are used to compare student results in different tests. Non-parametric tests are used for data which doesn't meet the requirements for using parametric tests.

RESULTS AND DISCUSSION

Results are presented in this section.

Problem-solving test

The first test was a problem-solving test that was administered before any programming lectures were held. The maximum test score was 14 points. Based on the achieved score, students were placed in one of three groups: stronger, intermediate and weaker students. The test can be seen in our previous paper (Mladenović, Krpan, et al., 2016). Table 2 shows distribution of participants by strength groups.

Table 2: Distribution of participants by strength groups

Group	N	points	Mean	SD
Stronger	15	>=11	11,93	1,223
Intermediate	16	Between 7 and 11	9,69	0,704
Weaker	19	<8	4,05	2,97

Post-tests

Two post-tests were conducted in order to assess students' achievement. The first post-test was administered following the conclusion of Python lectures, and the second one following conclusion of Scratch lectures. Since the number of points in each test was different, we decided to use the percentage as a measure of success. Kolmogorov-Smirnov test showed that there is a normal distribution of data in both Python (p=0,197) and Scratch (p=0,069) tests, but not in all groups combined. Table 3 shows descriptive statistics results.

Table 3: Descriptive statistics

	Python test			Scratch test		
	Mean	**SD**	**Shapiro-Wilk p**	**Mean**	**SD**	**Shapiro-Wilk p**
Stronger	81,667	20,5116	0,006	76,953	19,7003	0,095
Intermediate	61,831	22,5395	0,278	71,575	19,0275	0,253
Weaker	38,910	21,9488	0,504	54,537	17,1179	0,505

Because the results of the Python test didn't satisfy the assumption of normal distribution, the Kruskal-Wallis test was conducted. The test showed statistically significant difference between strength groups ($\chi^2(2)=19.342$, p=0.000). As it can be seen in Table 3, stronger students achieved, statistically significant, better results compared to intermediate and weaker.
ANOVA test was conducted to compare group results in Scratch test. There was a statistically significant difference between groups as determined by one-way ANOVA ($F(2,47) = 6.943$, $p = .002$). As ANOVA showed statistically significant difference, we made further analysis by Man-Whitney U and independent t-test differences between groups whose results are presented in the following sections.

Comparing success based on problem-solving abilities

In order to compare student success based on programming language used, we used the Mann-Whitney U test. In the analysis, we considered only two-thirds of the participants, those with intermediate and higher problem-solving skills. Students with lower problem-solving skills were left out of the analysis. When the Mann-Whitney statistic was calculated to determine whether there was any statistically significant difference in the Python test scores (U =62,5, z=-2,288, p=0,022), a statistically significant difference was found between students with higher problem-solving skills and those with intermediate problem-solving skills. From these results, we conclude that the former group of students achieved better results than the latter. In the case of Scratch test scores, no statistically significant difference between groups was found (U =99,5, z=-0,828, p=0,408).
These results indicate that with programming language like Scratch we can boost “middle-third students” for programming. This finding is consistent with other studies which showed that “middle third” students have the most benefit from the use of animations (Ben-Bassat Levy et al., 2003) and visual programming languages like Scratch (Armoni et al., 2015).
Afterwards, we compared intermediate and weaker students by independent t-test for both test results. Results of the t-test showed that students from intermediate group achieved statistically significant better results on the Python (t(33)=3,040, p=0,005), and Scratch test (t(33)=2.788, p=0.009). These results indicate that weaker students are “struggling” with programming regardless of the programing language used.
Based on the result we accept H1 because problem-solving abilities are directly related to success in Python programming language. But, when it comes to Scratch success this is not the case, at least for intermediate students so we can reject H2 because “middle third” students are equally successful in Scratch as stronger students.
Based on these results we can conclude that students with higher problem-solving skills can master programming regardless of the programming language or method used. An important finding is that by programming games in visual programming languages like Scratch we can stimulate the motivation of intermediate students. If we add a motivational factor, it’s worth to give a chance to new programming languages and approaches to reduce quitting from programming.

Attitude towards programming

In H3 we assumed that a positive attitude towards programming would be higher after using Scratch compared to Python. After the lessons about programming in Python, students answered a Likert scale question of 5 items about their attitude towards programming. This question was repeated in the small questionnaire students answered after the Scratch lessons. The questionnaire was composed of four Likert scale questions regarding their attitude towards the programming languages used.
Table 4 shows the questions.

Table 4: Survey questions

	Questions	
After Python	Q1	How much do you like programming?
After Scratch	Q2	How much do you like programming?
	Q3	How much do you like Python?
	Q4	How much do you like Scratch?
	Q5	Which programming language do you prefer?

Figure 1 shows frequencies.

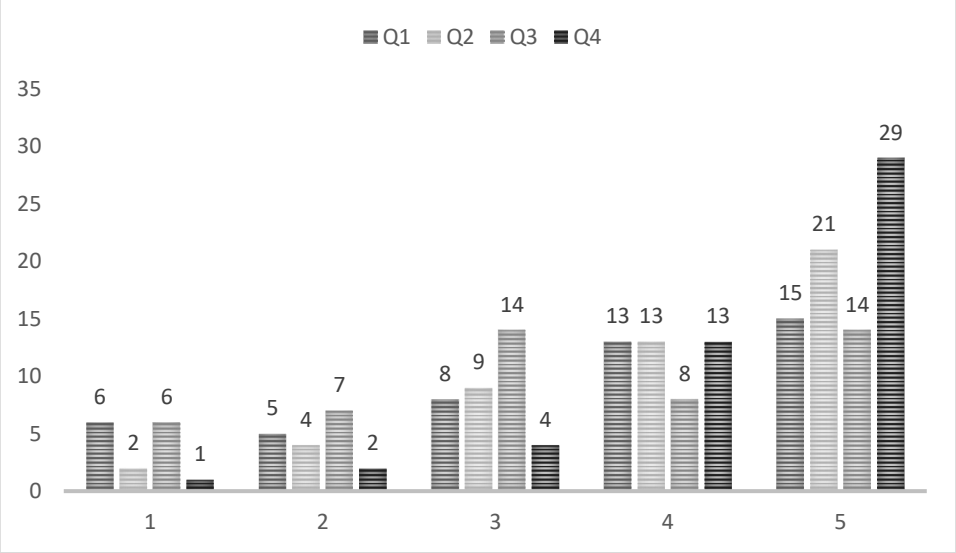

Figure 1: frequencies of the answers

From Figure 1 it is obvious that students liked Scratch more and that attitude towards programming is more positive after using Scratch. Figure 2 shows results for Q5.

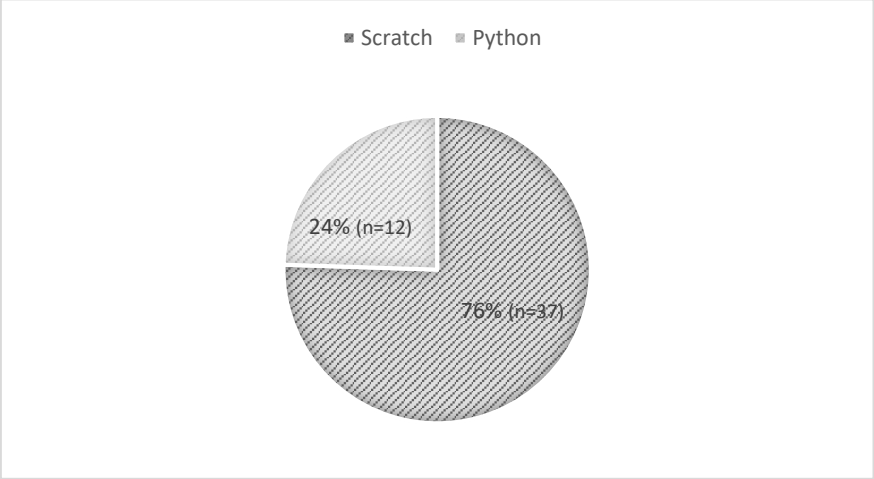

Figure 2: Q5 results

We wanted to compare student attitude towards programming after Python (Q1) and after Scratch (Q2). The Wilcoxon signed test rank showed statistically significant difference between Q1 and Q2 groups (Z=-2,012, p=0,044). Students had a greater affiliation for programming after Scratch compared to Python. This confirms that Scratch had a positive effect on student attitude towards programming. Thus we can accept H3 and conclude that attitude towards programming is more positive after Scratch than after Python.

However, it came as a surprise that even after being introduced to Scratch, a handful of students still preferred Python. We assumed that these are the students that belong to the top third of the class with regard to their problem-solving abilities.

In the questionnaire, students had space to write their thoughts about programming. They wrote 32 comments, only 4 of them were negative. For example:

- "*this is boring*", "*it's too hard for me*", '*it's too complicated*'.

Some of the comments were neutral:

- "*sometimes is boring and sometimes fun*" …

Most of the comments (22) about programming were positive:

- "*Programming is awesome*.", "*Programming is cool*.", "*Programming is great and interesting*.", "*I like programming, and I would like to learn it again next school year*.", "*I like programming because I learned something new*",

Some comments referring to Scratch:

- ”*I like Scratch more than Python*”; “*Scratch is extremely fun, it’s nice to see a game that I made*.”; “*I don’t like programming too much but I had fun while programming in Scratch*”; “*I like both Python and Scratch but I’ve chosen Python as favorite programming language because I was on Python programming competition*”; “*I like both programming languages, Python and Scratch*”; “*Scratch is awesome.*”…

Since the first author was the teacher in all classes, we can also confirm the observations (Armoni et al., 2015) which refer to early recognition of though concepts in a second programming language. Furthermore, we also observed a shortened teaching process which enables the teacher to assist students with weaker programming abilities, while those with higher programming abilities could explore new features in Scratch.

CONCLUSION

Programming novices, especially those at the elementary school level need a very gentle introduction to programming. Students should be able to focus on problem-solving and writing algorithms instead of thinking about syntax. Visual programming languages, like Scratch, offer the experience of syntax free programming which is suitable for novices. Furthermore, visual programming languages allow the teacher to shift the teaching context from solving math problems to programming games. Finally, is improves positive attitude towards programming. Considering statements above we need to be careful not to forget that the main reason for using visual programming languages is to focus on teaching programming concepts. Scratch should be a media or a tool used for transfer of those programming concepts into “real” text-based programming languages like Python.

Students with high problem-solving abilities can master programming easily, regardless of the programming language. On the other hand, students with lower problem-solving abilities encounter significant difficulties while learning programming, and might be unable to truly master it. These two groups usually constitute two-thirds of students in a class. The "middle third" students are the ones that we can influence the most. This is a group that is able to master programming with some difficulties. Based on some previous studies we assumed that the use of Scratch might boost their motivation, attitudes and achievement. With the teacher's help, Scratch can be used as a tool for mediated transfer of programming concepts from block-based to text-based programming languages and can improve the motivation for all students.

There is a lack of empirical research which compares the use of text-based and block-based visual programming languages in school settings at the K-12 level. We conducted research among 50 5th-graders in two elementary schools. Students were learning programming in Python, and later in Scratch. Results showed that students with higher problem-solving abilities were more successful in Python programming than students with lower problem-solving abilities. This is not the case when it comes to Scratch. In the case of Scratch, there were no differences in the success between better and “middle third” students which proves the usefulness of using Scratch to learn programming. Most students had more positive attitude towards programming after Scratch than after Python. It is important to note that students learned Python first, which is more difficult than Scratch. This order of introducing different programming languages might seem inversed, but we believe that it had a positive influence on the student's motivation. We believe that their motivation would be smaller if the languages were introduced vice versa.

REFERENCES

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From Scratch to “real” programming. *ACM Transactions on Computing Education*, *14*(4), 25:1–25:15. https://doi.org/10.1145/2677087

Ben-Bassat Levy, R., Ben-Ari, M., & Uronen, P. A. (2003). The Jeliot 2000 program animation system. In *Computers and Education* (Vol. 40, pp. 1–15). https://doi.org/10.1016/S0360-1315(02)00076-3

Cohen, L., Manion, L., & Morrison, K. (2013). *Research methods in education*. Routledge.

Dann, W., & Cooper, S. (2009). Alice 3: Concrete to Abstract. *Communications of the ACM*, *52*(8), 27. https://doi.org/10.1145/1536616.1536628

Dann, W., Cosgrove, D., Slater, D., & Culyba, D. (2012). Mediated Transfer : Alice 3 to Java. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 141–146. https://doi.org/10.1145/2157136.2157180

Gordon, M., Marron, A., & Meerbaum-Salant, O. (2012). Spaghetti for the main course?: observations on the naturalness of scenario-based programming. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education - ITiCSE ’12* (p. 198). New York, New York, USA: ACM Press. https://doi.org/10.1145/2325296.2325346

Grover, S., & Pea, R. (2013). Computational Thinking in K--12 A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43.

Maloney, J., Peppler, K., B. Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. *ACM SIGCSE Bulletin*, *40*(1), 367–371.

https://doi.org/10.1145/1352322.1352260
McCracken, M., Wilusz, T., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., … Utting, I. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In *Working group reports from ITiCSE on Innovation and technology in computer science education - ITiCSE-WGR '01* (p. 125). New York, New York, USA: ACM Press. https://doi.org/10.1145/572133.572137
Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2011). Habits of programming in Scratch. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education* (pp. 168–172). https://doi.org/10.1145/1999747.1999796
Ministry of science education and Sports of the Republic of Croatia. (2005). *The curriculum for primary school*. Zagreb.
Mladenović, M., Krpan, D., & Mladenović, S. (2016). INTRODUCING PROGRAMMING TO ELEMENTARY STUDENTS NOVICES BY USING GAME DEVELOPMENT IN PYTHON AND SCRATCH. In *EDULEARN16 Proceedings* (pp. 1622–1629). IATED. https://doi.org/10.21125/edulearn.2016.1323
Mladenović, M., Rosić, M., & Mladenović, S. (2016). Comparing Elementary Students ' Programming Success Based on Programming Environment. *International Journal of Modern Education and Computer Science*, *8*(August), 1–10. https://doi.org/10.5815/ijmecs.2016.08.01
Moreno, J., & Robles, G. (2014). Automatic detection of bad programming habits in scratch: A preliminary study. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings* (pp. 1–4). IEEE. https://doi.org/10.1109/FIE.2014.7044055
Pausch, R., & Zaslow, J. (2008). Last Lecture. *Statistics*, *7*, 1–18. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/19064375
Price, T. W., & Barnes, T. (2015). Comparing Textual and Block Interfaces in a Novice Programming Environment. In *Proceedings of the eleventh annual International Conference on International Computing Education Research - ICER '15* (pp. 91–99). New York, New York, USA: ACM Press. https://doi.org/10.1145/2787622.2787712
Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for All. *Commun. ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779
White, G., & Sivitanides, M. (2003). An Empirical Investigation of the Relationship Between Success in Mathematics and Visual Programming Courses. *Journal of Information Systems Education*, *14*(4).
Whitfield, A. K., Blakeway, S., Herterich, G. E., & Beaumont, C. (2007). Programming, disciplines and methods adopted at Liverpool Hope University. *Innovation in Teaching and Learning in Information and Computer Sciences*, *6*(4), 145–168.