THE ROLE OF NEW TECHNOLOGY IN EDUCATION OF PROGRAMMING NOVICES

Mario Konecki[1], Mladen Konecki[1] and Ivana Ogrizek Biškupić[2]
[1]*Faculty of Organization and Informatics, University of Zagreb*
Pavlinska 2, 42000 Varaždin, Croatia
[2]*Algebra University College*
Gradišćanska 24, 10000 Zagreb, Croatia

ABSTRACT

Constant efforts to develop more and better information systems have become a critical factor of successful functioning of virtually all major systems. The world has become highly dependent on well-designed and well-developed information infrastructure. Education systems are constantly struggling to provide sufficient number of information system professionals, especially programmers. However, there is a constant lack of this kind of professionals on the market, and one of possible reasons for this kind of state is the fact that education of programming novices has proven to be extremely challenging and complex task for their educators. In this paper a discussion about the challenges and reasons for these challenges in education of programming novices is given. Possible aiding approach and technological solutions that could impact student motivation and work habits are presented and elaborated.

KEYWORDS

Programming novices, Education, Challenges, Motivation, New technology

1. INTRODUCTION

It is very common for educators to state that it is very hard and challenging to teach programming novices how to create functional programming solutions. Teachers have tried various approaches to this day but the problem of teaching novices how to program has persisted.

Students tend to perceive programming as very hard and abstract content, and frequently they are simply not motivated enough to deal with all challenges. This kind of situation results in the general opinion that programming is hard (Baldwin Kuljis, 2001; Bergin and Reilly, 2005; Gomes and Mendes, 2007; Hanks et al., 2004; Jenkins, 2002; Peng, 2010; Robins et al., 2003) but what is significantly more important is that this kind of situation results is quite high failure and dropout rates (Nikula et al., 2011; Yadin, 2011).

This problem itself is of high importance because lower education success rate results in lower number of highly skilled programming experts that will be able to continue with the rapid development of information systems that are of vital importance for virtually all major real-life systems today.

In order to really understand this problem, it is of great importance to understand all the factors that influence this kind of situation. On one side there are teachers with their own teaching preferences, technological and pedagogical knowledge and skills, characters and teaching approaches.

On the other side there are students who also have their individual learning preferences, characteristics, prior knowledge and skills regarding technology, programming and problem-solving. It is important to not only understand teachers but also students and the interaction that the education process implies. Another aspect that needs to be considered is the nature of the learning subject itself.

Programming is a specific learning matter that is abstract and complex. Because of this it requires a prolonged way of learning, and a lot of practical experience. This learning strategy is in many cases something that students are not used to because of different learning strategy that can be used on most of their other courses.

This means that it is even more time-consuming and energy-consuming for students to adapt to a learning style required to master programming as a skill. In this aspect motivation plays a major role because only

motivated students will be willing to invest additional effort to learn this complex skill. Other research has also shown that motivation is of high importance for successful programming education (Kinnunen and Malmi, 2006).

2. CHALLENGES AND POSSIBLE APPROACHES

Many different challenges and questions can be stated when talking about programming education on both teachers' and students' side.

On the teachers' side some of the challenges and questions are:

- What programming language is best suited for programming novices?
- Is the popularity and the level of possible professional application of the programming language of greater importance than the novice-oriented style of programming language?
- Are students' prior skills and knowledge prerequisite for successful programming education, what skills and knowledge, and to what extent?
- Is "one size fits all" approach to teaching programming a suitable approach to education of programming novices?
- What is the best pedagogical approach to teaching programming novices?
- Is programming course content simply too much for some students, and should some elements be reduced?
- How to use technology to help students understand abstract and complex programming concepts?
- How to use technology to motivate students to deal with programming?
- How much practical work should be incorporated into the programming course?
- How to support development of problem-solving skills?
- etc.

As it can be seen, teachers have more than complex situation regarding decisions and effort they need to invest in order to create the best possible educational ecosystem for programming novices.

The situation is equally challenging on the students' side. Some of the challenges and questions are:

- How much programming practice is enough?
- What learning strategy will work best?
- How to visualize and understand abstract programming concepts?
- How to gain competence in problem-solving skills?
- How to manage learning time considering other courses?
- How to overcome fear and anxiety regarding programming?
- How to stay motivated and focused?
- etc.

As can be seen, the students face as much challenges as their teachers. Research has shown that fear of programming is one of key aspects that can form initial attitude of programming novices toward their programming courses (Rogerson and Scott, 2010).

One of the problems is that research has shown that many students who enroll programming courses simply do not have sufficient interest in programming activities. Research has shown the following (Bergin and Reilly, 2005):

- Only 22% of students enroll in programming courses because of interest.
- 40% of students enroll in programing courses because they think it is necessary to do so in order to have a prospective career options.
- As much as 35% of students enroll in programming courses just because they have no choice since these are mandatory courses.

As can be seen, the fact that so many students enroll programming courses not because of genuine interest but because of other reasons is a huge additional challenge for their programming educators. Teachers can use extrinsic motivation measures, but research has shown that intrinsic motivation is of even greater importance (Nikula et al., 2011) and this is something that is very hard and complex to influence.

Teachers have tried to use many programming languages in their search for the most suitable one for novices, and there are many programming languages that have been designed with the programming novices in mind, such as (Smith et al., 2000): Logo, Basic, Pascal, Smalltalk, and HyperTalk.

However, some authors claim that the programming language that would be completely suitable for programming novices does not exist and will never exist (Smith et al., 2000).

Research has shown that one of the problems is the way students are used to learn since most of their former education was based on memorization of facts, not adopting a new skill such as programming (Gomes and Mendes, 2007). This has created a gap between the way programming novices think and the way of thinking needed to understand algorithms.

Some authors claim that this gap is as wide as Grand Canyon (Norman and Draper, 1986). Many students try to learn programming not in a prolonged repetitive manner but in a short period of time (Lister et al., 2004). This results in gaining knowledge about the programming language structure but does little to increase skill level in the domain of algorithmic thinking and problem-solving skills.

When talking about possible solutions for challenges in education of programming novices then many different approaches can be identified (Konecki, 2014):

- Introduce additional programming course prior to introductory programming course that would promote algorithmic way of thinking
- Increase motivation of students for learning programming
- Explain to students that programming is a skill, not merely knowledge
- Introduce elements of constructivism into teaching process
- Introduce learning by example
- Introduce animation and other visualization techniques combined with interaction
- Introduce interactive visual simulations
- Include support for multiple learning styles

As can be seen, many different approaches can be taken but to this day no final solution has been identified. Whether this situation is persisting because of not enough measures being taken or whether the right measure has not yet been invented is something that is yet to be researched and concluded about.

3. NEW TECHNOLOGY IN EDUCATION

One of possible approaches to aiding programming novices in their programming education is usage of new technology to make abstract programming concepts clearer and more comprehensive to students, and what is of great importance is to try increasing students' motivation to learn programming. This is done by giving students tools that are of educative nature but also serve as examples of what can be achieved by development activities. There are many different technologies that can be used in education, and the same is true for the education of programming novices.

Some of already well-known and used technological solutions include: video materials, chats, forums, quizzes, self-assessment tools, presentations, various reading learning content, educative visual programming environments, such as Alice, Kodu, Scratch, and Greenfoot (Jahongir, 2022), visual programming languages (Tsai, 2019).

Interactive animations are also well-known but are less used than formerly mentioned content types.

Some of less known and less-commonly used technological solutions that represent latest forms of learning content are: virtual reality (Christou, 2010), artificial intelligence (Roll and Wylie, 2016), educative videogames (Lee et al., 2004), and gamified educational content (Caponetto et al., 2014).

Potential advantages of mentioned new technologies are: better visualization and representation options of various programming concepts, higher level of immersion and engagement in educative content, and multi-sensory experience of various educational elements.

4. METHODOLOGY AND RESEARCH RESULTS

In order to conclude about possible impact of new technology a research that included 87 information technology students who participated in the research has been conducted. All students have been given three new technology-based educational systems that included the following technologies:

- Virtual reality
- Artificial intelligence
- 3D modeling

Virtual reality has been incorporated in the form of custom virtual reality learning environment (shown in Figure 1) in which students were able to experience different programming concepts in the form of implemented videogame, and they were also able to test their knowledge through developed virtual reality self-assessment module.

Artificial intelligence has been incorporated in the form of virtual assistant (Konecki et al., 2015; Konecki and Kadoić, 2015) (shown in Figure 1) that enabled students to ask different questions related to programming. Virtual assistant has provided answers, along with additional learning content related to the question.

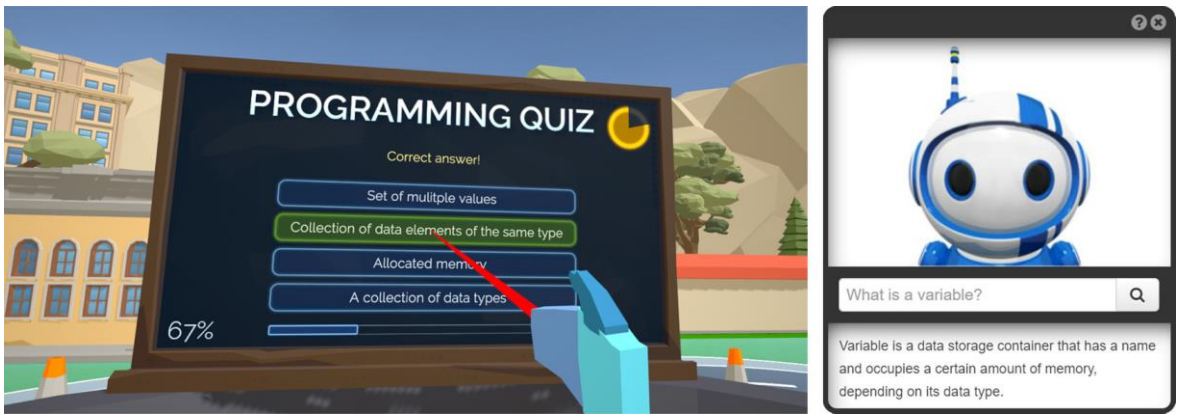

Figure 1. Virtual reality learning environment and virtual assistant

3D modeling has been incorporated in the form of virtual 3D realistic model of the faculty (shown in Figure 2). By using this interactive 3D model students were able to walk through the faculty, and to learn about its structure. Developed interactive 3D model served as a demonstration of what can be achieved by using programming knowledge and skills, but it also provided students with a fun and engaging way to learn about the faculty.

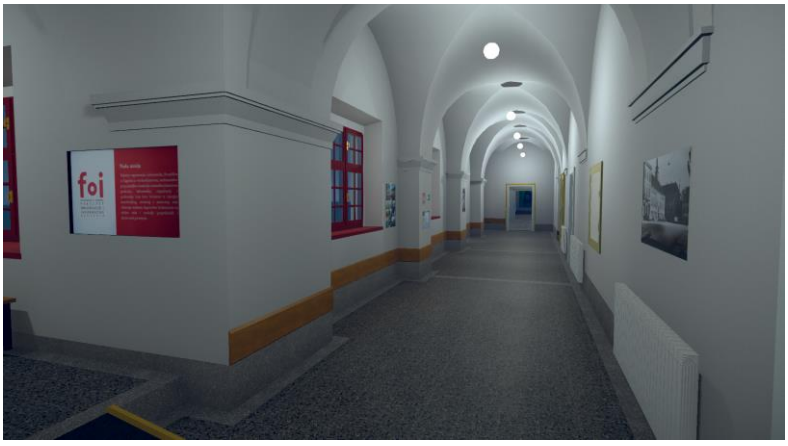

Figure 2. 3D realistic model of the faculty

After mentioned experiences a research in the form of questionnaire has been conducted to assess whether this new technology experience has affected students' motivation and whether they feel more ready to do programming exercises in a more prolonged manner compared to before. In order to do so, research has been conducted in two parts in order to conclude about the difference in their perception and motivation. Likert scale was used ranging from "Strongly disagree" (1) to "Strongly agree" (5). Research results are shown in Table 1 and Table 2.

Table 1. Research results (Students' attitude towards programming education)

R-1.1	Statement	$\bar{x}$	SD
1.	I am motivated to learn programming	3,11	1,05
2.	I am willing to learn programming in a prolonged way	3,01	1,14
3.	I am willing to do all programming exercises that are available to me as a part of my programming course each week	2,95	1,12
4.	I would like to know programming better	3,25	1,00

Results of the first part of the research have shown that students are moderately motivated to learn programing and to put additional effort needed to learn programming as a skill. Slightly more affirmative was the statement about wanting to know programming better.

Table 2. Research results (Students' attitude towards programming education)

R-1.2	Statement	$\bar{x}$	SD
1.	I am motivated to learn programming	3,41	1,02
2.	I am willing to learn programming in a prolonged way	3,32	1,12
3.	I am willing to do all programming exercises that are available to me as a part of my programming course each week	3,30	0,98
4.	I would like to know programming better	3,62	0,82

After the second part of the research was conducted, a paired samples t-test was performed to compare the strength of agreement with the four statements before and after students' experience with new technology in programming education.

There was a significant difference (p=0.015) in being motivated to learn programming before (M = 3.09, SD = 1.06) and after students' experience with new technology in programming education (M = 3.39, SD = 1.04).

The respondents on average were more willing to learn programming in a prolonged way (p=0.026) after engaging in experience with new technology in programming education (M = 3.29, SD = 1.12) than before (M = 2.99, SD = 1.14).

Also, the respondents on average were more willing to do all programming exercises that are available as a part of the programming course each week (p=0.009) after experience with new technology in programming education (M = 2.93, SD = 1.12) than before (M = 3.27, SD = 0.99).

The respondents on average agreed more (p=0.001) with the statement „I would like to know programming better" after experience with new technology in programming education (M = 3.23, SD = 1.02) than before (M = 3.59, SD = 0.87).

5. CONCLUSION

Programming is one of vitally important professions, especially in the context of the modern world and society. However, both programing novices and their teachers experience various difficulties and challenges in their programing courses and education.

In this paper, different challenges that both programing novices and teachers experience in programming education have been presented and discussed, along with the reasons for these challenges. Possible approaches and solutions have also been presented and elaborated. Research results about the impact of new technology on the motivation of programing novices and their working habits have been presented and analyzed.

New technology has proven to be a promising mean of aiding programing novices in their programming education, especially in raising their motivation to learn programming as a skill. More research is needed to conclude about whether these effects will be long lasting and whether the course results will follow the perception and impression of programming novices. This part will be the focus of the future research efforts.

REFERENCES

Baldwin, L. P. and Kuljis, J., 2001. Learning programming using program visualization techniques. *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, Washington, DC, USA, pp. 1051-1058.

Bergin, S. and Reilly, R., 2005. The influence of motivation and comfort-level on learning to program. *Proceedings of the 17th Annual Workshop on the Psychology of Programming Interest Group*, Brighton, UK, pp. 293-304.

Caponetto, I. et al., 2014. Gamification and education: A literature review. *European Conference on Games Based Learning, Academic Conferences International Limited*, Vol. 1, p. 50.

Christou, C., 2010. Virtual reality in education. *Affective, interactive and cognitive methods for e-learning design: creating an optimal education experience*, pp. 228-243.

Gomes A. and Mendes, A. J., 2007. An environment to improve programming education. *Proceedings of the 2007 international conference on Computer systems and technologies*, New York, USA, pp. 1-6.

Hanks, B. et al., 2004. Program quality with pair programming in CS1. *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, United Kingdom, pp. 176-180.

Jahongir, Z., 2022. The Role of Visual Learning in Improving Students' High-Order Thinking Skills. *Barqarorlik va Yetakchi Tadqiqotlar onlayn ilmiy jurnali*, 2(2), pp. 252-256.

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

Kinnunen, P. and Malmi, L., 2006. Why students drop out CS1 course?. *Proceedings of the 2006 International Workshop on Computing Education Research*, pp. 97-108.

Konecki, M., 2014. Problems in programming education and means of their improvement. *DAAAM international scientific book*, pp. 459-470.

Konecki, M. et al., 2015. MIA: A multi-purpose intelligent assistant. *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1109-1113.

Konecki, M. and Kadoić, N., 2015. Intelligent assistant for helping students to learn programming. *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 924-928.

Lee, J. et al., 2004. More than just fun and games: Assessing the value of educational video games in the classroom. *CHI'04 extended abstracts on Human factors in computing systems*, pp. 1375-1378.

Lister, R. et al., 2004. A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, 36(4), pp. 119–150.

Nikula, U. et al., 2011. A motivation guided holistic rehabilitation of the first programming course. *ACM Transactions on Computing Education (TOCE)*, 11(4), 24.

Norman D. A. and Draper, S. W., 1986. *User-Centered System Design: New Perspectives on Human Computer Interaction*, Erlbaum, Hillsdale, NJ.

Peng, W., 2010. Practice and experience in the application of problem-based learning in computer programming course. *Proceedings of the International Conference on Educational and Information Technology (ICEIT)*, pp. 170-172.

Robins, A. et al., 2003. Learning and Teaching Programming: A Review and Discussion. *Journal of Computer Science Education*, 13(2), pp. 137-172.

Rogerson C. and Scott, E., 2010. The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education: Research*, 9(1), pp. 147-171.

Roll, I. and Wylie, R., 2016. Evolution and revolution in artificial intelligence in education. *International Journal of Artificial Intelligence in Education*, 26, pp. 582-599.

Smith, A. et al., 2000. Programming by example: novice programming comes of age. *Communications of the ACM*, 43(3), pp. 75-81.

Tsai, C. Y., 2019. Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy. *Computers in Human Behavior*, 95, pp. 224-232.

Yadin, A., 2011. Reducing the dropout rate in an introductory programming course. *ACM Inroads*, 2(4), pp. 71-76.