Uluslararası İleri Doğa Bilimleri ve Mühendislik Araştırmaları Dergisi
Sayı 9, S. 212-218, 3, 2025

Araştırma Makalesi

IJANSER
https://as-proceeding.com/index.php/ijanser
ISSN:2980-0811

International Journal of Advanced Natural Sciences and Engineering Researches
Volume 9, pp. 212-218, 3, 2025

Research Article

The importance of using visualization and simulation in teaching programming

József Udvaros[*1,2]

[1]*Department of Mathematics and Computer Science, Trnava University, Slovakia*
[2]*Department of Business Information Technology, Budapest University of Economics and Business, Hungary*

[*]*(jozsef.udvaros@truni.sk) Email of the corresponding author*

(Received: 04 March 2025, Accepted: 07 March 2025)

(4th International Conference on Recent Academic Studies ICRAS 2025, March 04-05, 2025)

ATIF/REFERENCE: Udvaros, J. (2025). The importance of using visualization and simulation in teaching programming. *International Journal of Advanced Natural Sciences and Engineering Researches*, 9(3), 212-218.

Abstract – Visualization and simulation techniques are increasingly used in computer science and teaching programming to help students gain a deeper understanding and develop their problem-solving skills. Learning the basics of programming is often difficult for novice students, especially when it comes to understanding abstract concepts such as algorithms, data structures, and control structures. The use of visual tools and interactive simulations provides an opportunity to put theoretical knowledge into practice, thereby reducing cognitive load and increasing motivation.
The study presents the benefits of using visualization and simulation methods in teaching programming, with a particular focus on algorithm visualization, block-based programming environments, and simulation models. During the research, we analyzed various tools and platforms that can help students learn the fundamentals of programming more easily. The results show that visual learning strategies contribute to the development of problem-solving skills and the practical application of theoretical knowledge. The study aims to provide guidance for programming educators on how to effectively incorporate modern technologies into the educational process.

Keywords – Visualization, Simulation, Teaching Programming, Algorithm Visualization, Educational Process.

I. INTRODUCTION

Teaching programming is a fundamental pillar of computer science education, essential for developing students' algorithmic thinking and acquiring digital competences. However, learning and teaching programming comes with many challenges, especially for beginners. Understanding abstract concepts such as algorithms, data structures, and control structures can be difficult, often leading to frustration and loss of motivation [1].

Visualization and simulation are pedagogical tools that can significantly facilitate the teaching and learning of programming. Visualization helps students understand the logic behind lines of code and the functioning of algorithms, while simulations create opportunities to model real-world problems and gain practical experience. Interactive educational tools, such as algorithm visualization software and block-based programming environments, can help in the practical application of theoretical knowledge, thereby making the learning process more efficient [2].

The aim of this study is to present the importance of the use of visualization and simulation in teaching programming and to examine their impact on students' understanding, problem-solving skills and motivation. During the analysis, various educational tools and methods are presented that can facilitate more effective learning of programming. The results of the research can contribute to the development of modern educational strategies aimed at improving the learning experience of students and innovative approaches to teaching programming.

II. THEORETICAL BACKGROUND OF VISUALIZATION AND SIMULATION

During teaching programming, students often encounter abstract concepts, the understanding of which can represent a significant cognitive load. Traditional, textual presentation of the operation of algorithms, the behavior of data structures and the process of program execution is often not sufficient to develop a deeper understanding. Visualization and simulation are pedagogical approaches that help students better grasp these complex concepts, thereby facilitating the acquisition of programming skills.

A. The role of visualization in learning

Visualization is one of the most effective learning tools, allowing abstract concepts to be made more concrete. According to educational psychology research, visual information processing is faster and more efficient than purely verbal or textual learning. In teaching programming, visualization can take various forms:

- Algorithm visualization: Step-by-step visualization of the execution of algorithms helps students understand the operation of control structures.
- Code flow visualization: Showing the effect of individual instructions allows students to track the operation of programs.
- Data structure visualization: Illustrating how data is stored and processed can help in understanding abstract structures [3].

Visualization not only supports the acquisition of theoretical knowledge, but also helps in the development of debugging and problem-solving skills. Interactive visualization tools like Python Tutor or Visualgo give students the opportunity to experiment with running code and its effects at their own pace.

B. The Importance of Simulation in Teaching programming

Simulation techniques allow students to test their theoretical knowledge in a real or realistic environment. Simulations can be particularly useful in teaching programming in the following areas:

- Problem-based learning: Simulations allow students to learn programming principles by solving real-life problems.
- Modeling and testing: Simulating complex systems allows testing the effectiveness of different algorithms and programming solutions.
- Virtual labs: Online simulation platforms, such as CodinGame or MIT App Inventor, provide students with the opportunity to develop their programming skills in an interactive way.

Simulations can be particularly effective in teaching robotics, artificial intelligence, and IoT (Internet of Things), where the immediate impact of programming code can be observed visually and interactively. Modeling real-world environments and gaining practical experience contribute to deeper understanding and the development of independent problem solving [4][5][6].

C. Connection to constructivist pedagogical theories

The use of visualization and simulation methods fits well with the constructivist pedagogical approach, according to which learning is an active, self-directed process. Students are not just passive recipients of information, but build their knowledge through their own experiences and interactions. Interactive programming environments and visual tools facilitate independent discovery, experimentation, and practical application, which results in a more effective learning process in the long run.

Visualization and simulation are therefore not only complementary tools in teaching programming, but also play a key role in deepening students' understanding and increasing the efficiency of the learning

process. In the following chapters, we will present in detail the different visualization and simulation tools and their practical application in teaching programming.

III. VISUALIZATION TOOLS IN TEACHING PROGRAMMING

The effectiveness of teaching programming can be significantly increased by visualization tools that help in understanding algorithms, data structures, and programming concepts. These tools provide students with the opportunity to visually follow the operation of the code, thus reducing learning barriers and increasing motivation. In this chapter, we present the most commonly used visualization tools and their advantages [7].

A. Algorithm visualization

Understanding the operation of algorithms is one of the biggest challenges in learning programming. Algorithm visualization is a collection of techniques and tools that allow for the step-by-step presentation of the operation of algorithms. These tools help students to visually follow the execution of individual instructions and their effect on the data [8].

Examples of algorithm visualization tools:

- Python Tutor: An interactive tool that allows students to observe the execution of Python, C, C++, Java, and JavaScript code step-by-step.
- VisuAlgo: An online platform that specializes in visualizing various algorithms (e.g., sorting algorithms, graph algorithms).
- Algomation: A simple and user-friendly algorithm visualization tool that helps students understand the logic of algorithms.

Algorithm visualization is especially useful for beginning students, as it helps them apply theoretical concepts to practice and develop algorithmic thinking [9].

B. Block-based programming environments

Beginner programmers often have difficulty avoiding syntax errors and understanding code structure. Block-based programming environments offer a visual programming approach that reduces the frustration of syntax errors and allows for easier understanding of logical relationships.

Popular block-based programming environments:

- Scratch: A visual programming language developed by MIT that allows you to build algorithms and programs using colored blocks.
- Blockly: A tool developed by Google that offers an interactive, drag-and-drop programming experience.
- Snap!: An enhanced version of Scratch that is also suitable for creating more complex programs and mathematical models.

The advantage of block-based programming environments is that they help students learn basic programming concepts (e.g., loops, branches, variables) without having to deal with complex syntaxes right away [10].

C. Visual support in code editors and development environments

Choosing the right development environment (IDE) is crucial when learning programming, as these tools provide students with a variety of visual supports. Interactive editors, debuggers, and code highlighters help students understand the logical structure of code and quickly identify errors.

Popular development environments with visual features:

- Jupyter Notebook: An interactive development environment that allows you to run and visually represent Python code.
- Thonny: A Python development environment optimized for beginners that supports learning with built-in visualization and step-by-step debugging.
- Visual Studio Code: A popular code editor that supports interactive learning and visualization with a variety of plugins.

These tools make it easier to understand the structure and operation of code, and make the debugging process more efficient, which significantly contributes to the development of programming skills [11].

Visualization tools play a significant role in teaching programming, as they help illustrate theoretical concepts and deepen students' understanding. Algorithm visualization platforms, block-based programming environments, and development tools all contribute to more effective and motivated learning for students.

IV. SIMULATIONS IN PROGRAMMING LEARNING

Simulation techniques are increasingly playing a role in teaching programming, as they provide students with the opportunity to experiment and gain practical experience in realistic environments. Simulations are particularly useful in areas where programming is directly related to the operation of physical or virtual systems, such as robotics, network simulations, or data modeling.

A. The Role of Simulations in Teaching programming

Simulations play a prominent role in teaching programming, as they provide students with the opportunity to try out algorithms and programming solutions in an interactive and realistic environment. One of the most important advantages of simulations is that they reduce cognitive load, as students can visually follow the code in action and see the results in real time. This is particularly useful in understanding abstract concepts such as data structures, control structures, and algorithms, which are challenging for many students.

The use of simulations is particularly effective in terms of gaining practical experience, as students not only acquire theoretical knowledge, but also deepen their knowledge by modeling and solving real problems. Such tools help develop algorithmic thinking and problem-solving skills, while providing opportunities for experimentation and trying out creative solutions. Virtual labs and interactive programming environments allow students to perform tests in a safe environment without damaging real systems, which is especially important in the case of network simulations, robotics or embedded systems.

Increasing motivation is also an important factor in teaching programming, and simulations play a significant role in this. Realistic situations and interactive learning experiences contribute to student engagement and maintaining interest. Playful learning environments, such as gamified programming platforms, provide opportunities for students to master the basics in an enjoyable way [12].

Incorporating simulations into teaching programming can significantly enhance the effectiveness of the learning process. Interactive simulation tools help students understand theoretical concepts more easily, develop practical skills, and gain greater autonomy in problem-solving. However, for simulations to be truly useful, they require an appropriate teaching strategy and conscious integration into the curriculum that takes into account the different needs and abilities of students [13][14].

B. Programming Physical Systems with Simulations

Simulation tools that model physical systems, such as robots, sensors, or embedded systems, are becoming increasingly popular in teaching programming. These tools are particularly useful in engineering and computer science studies, as they provide students with the opportunity to test their codes in a real-world environment.

Popular robotics and embedded systems simulation tools include:

- Tinkercad Circuits – An online simulation platform that allows for modeling and programming of electronic circuits and embedded systems [15].
- Webots – An advanced robotics simulator that allows for programming and testing of complex robotic systems.
- VEXcode VR – A virtual robotics simulator that allows students to try out different algorithms in an interactive environment.

These tools allow students to model real-world problems and develop algorithms without the need for real hardware [16].

C. Virtual Labs and Interactive Coding Platforms

Virtual labs are interactive learning environments that allow students to learn programming concepts through online simulations. These platforms often provide real-time feedback to students and allow them to test their code in different environments.

Popular virtual labs and simulation platforms:

- CodinGame – An interactive programming platform that presents programming tasks and algorithms in a playful way.
- MIT App Inventor – A visual programming environment that allows you to create and simulate mobile applications.
- Google Colab – A cloud-based development environment that allows you to run Python scripts and perform visualization analyses.

These tools help develop programming skills by allowing students to directly test and modify their code in an interactive environment [17].

D. Simulations in learning algorithms and data structures

Simulations are useful not only for physical systems and virtual labs, but also for teaching algorithms and data structures. Interactive simulations help students understand how each algorithm works by visually following the different steps.

Algorithm and data structure simulation tools:

- Sorting Algorithms Visualizer – An online tool that allows visualizing the operation of various sorting algorithms.
- Graph Theory Playground – An interactive simulation environment that facilitates the learning of graph algorithms.
- Heap Visualizer – A tool that allows visual understanding of priority queues and heap data structures.

These types of tools help students to understand complex algorithms and data structures more easily and apply them more effectively [18].

Simulation tools play a key role in teaching programming, as they provide students with the opportunity to gain hands-on experience in real or virtual environments. Interactive robotics simulators, virtual labs, and algorithm simulation platforms all contribute to the effectiveness of teaching programming. In the next section, we present empirical examples and research findings that support the effectiveness of visualization and simulation in teaching programming [19].

V. DISCUSSION

The use of visualization and simulation in teaching programming has many advantages, but it is not without challenges. One of the biggest advantages is that it helps students understand abstract concepts and develop algorithmic thinking. Interactive visualization tools and simulation platforms reduce cognitive load, provide students with the opportunity to actively engage in the learning process, and gradually master the fundamentals of programming. In addition, real-world simulations provide students with hands-on experience, which in the long term increases the employability of their programming skills.

However, the introduction of visualization and simulation methods into education also poses technological and pedagogical challenges. Operating such tools is often resource-intensive, and not all educational institutions have the necessary hardware or software. In addition, teachers also need to be familiar with and effectively use these tools, which requires appropriate training. Another important issue is that excessive reliance on visualization can hinder students' development in text-based programming environments, which are essential for developing more advanced programming skills. Therefore, visualization and simulation techniques should be properly integrated into the teaching process so that they support, but do not replace, the thorough mastery of programming.

VI. CONCLUSION

The use of visualization and simulation can be a significant step forward in teaching programming, especially for beginning students. These tools help to better understand algorithms and data structures, increase student motivation, and provide opportunities for independent experimentation. Interactive learning environments and modelling real-world applications can contribute to strengthening the practical side of programming, which is beneficial for both education and subsequent professional development.

In order for these techniques to effectively support learning, it is necessary to develop pedagogical methods and educational infrastructure. The gradual introduction of visualization and simulation, strengthening teacher training, and providing appropriate hardware and software background are key factors. In addition, it is important that such tools do not become exclusive, but act as complementary tools, facilitating the development of students' independent coding skills. If you can strike the right balance between visualization and traditional text-based programming, teaching programming can become more effective, understandable, and enjoyable, resulting in deeper knowledge and better problem-solving skills in the long run.

ACKNOWLEDGMENT

Financial support for this study was provided from the project KEGA 014TTU-4/2024 “Intelligent Animation-Simulation Models, Tools, and Environments for Deep Learning”.

REFERENCES

[1] Gabaľová, V., Lengyelová, A., Oppenberger, H.V. and Stoffová, V., 2023. ICT in children's compulsory pre-school education. In EDULEARN23 Proceedings (pp. 2334-2338). IATED.
[2] Czakóová, K., Takáč, O. and Végh, L., 2023. Program Code Simulation by Using Ozobot. International Journal of Advanced Natural Sciences and Engineering Researches, 7(10), pp.441-444.
[3] Stoffová, V. and Gabaľová, V., 2023. Creation Of Electronic Teaching Aids And Didactic Applications By Teachers. In ICERI2023 Proceedings (pp. 5344-5353). IATED.
[4] Dancsa, D., Štempeľová, I., Takáč, O. and Annuš, N., 2023. Digital tools in education. International Journal of Advanced Natural Sciences and Engineering Researches, 7(4), pp.289-294.
[5] Gabaľová, V., 2023. Microworlds as a first step in teaching programming. In EDULEARN23 Proceedings (pp. 2329-2333). IATED.
[6] Hyksova, H., Stoffová, V. and Gabaľová, V., 2022. Teaching robotics and online programming using a virtual robot. In INTED2022 Proceedings (pp. 7140-7147). IATED.
[7] Végh, L. and Stoffová, V., 2017. Algorithm animations for teaching and learning the main ideas of basic sortings. Informatics in Education, 16(1), pp.121-140.
[8] Singh, A.P., 2024. Sorting and Path Finding Algorithm Visualizer. International Journal on Smart & Sustainable Intelligent Computing, 1(2), pp.40-48.
[9] Pšenáková, I. and Baganj, I., 2016. Možnosti využitia prostriedkov virtuálneho sveta vo vzdelávaní. Edukacja-Technika-Informatyka, 7(1), pp.212-218.
[10] Czakóová, K. and Takáč, O., 2022. Interactive Programming Tools for Beginners in Elementary School Informatics. In EDULEARN22 Proceedings (pp. 5588-5595). IATED.
[11] Bunn, T., Anslow, C. and Lundqvist, K., 2024. Towards a Python 3 processing IDE for teaching creative programming. Multimedia Tools and Applications, 83(38), pp.86247-86260.
[12] Pšenaková, I. and Szabó, T., 2018, November. Interactivity in learning materials for the teaching. In 2018 16th International Conference on Emerging eLearning Technologies and Applications (ICETA) (pp. 445-450). IEEE.
[13] Stoffová, V. and Gabaľová, V., 2023. Interactive Simulation Models for Teaching and Learning of Computer Science. In INTED2023 Proceedings (pp. 3945-3953). IATED.
[14] Gabaľová, V. and Stoffová, V., Programovanie v prostredí Lazarus na stredných školách. Przygotowanie nauczycieli do nowych wyzwań edukacyjnych: Problemy współczesnej edukacji, pp.158-167.
[15] Udvaros, J. and Forman, N. (2024). Tinkercad as a virtual laboratory. In Proceedings of the International Conferences on e-Learning and Digital Learning 2024, Sustainability, Technology and Education 2024 (pp. 91–97).
[16] Abonyi-Tóth, A., 2021. Possibilities of Simulating Robot Generations in Public Education. Central-European Journal of New Technologies in Research, Education and Practice, pp.1-19.
[17] Eguíluz, A., Garaizar, P. and Guenaga, M., 2018. An evaluation of open digital gaming platforms for developing computational thinking skills. Simulation and gaming, 10.

[18] Korhonen, A. and Malmi, L., 2000, July. Algorithm simulation with automatic assessment. In Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSEconference on Innovation and technology in computer science education (pp. 160-163).
[19] Végh, L. and Czakóová, K., 2023. Possibilities of using games in teaching and learning the basic concepts of object-oriented programming. In INTED2023 Proceedings (pp. 5329-5334). IATED.