PROGRAMMING IN PRIMARY EDUCATION AND BUILDING 21ST CENTURY COMPETENCIES

Georgi Hristov

Abstract. *This paper offers an overview of the original concept of introducing computer programming into the classroom. The key features of Scratch, currently the most widely used programming language for teaching elementary students, are outlined. Additionally, the factors contributing to Scratch's growing popularity are examined, along with its role in fostering students' computational thinking. Recommendations are made for integrating Scratch into new educational contexts, and some of the most significant challenges impeding the teaching of programming to young learners are identified.*

The relevance and role of programming-related curricular topics for students, in the context of developing 21st-century competencies, are explored. In this regard, the European Digital Competence Framework for Citizens (DigComp 2.2) is reviewed, with particular attention to the programming competency. Examples from DigComp 2.2 are provided to illustrate how this competence manifests in terms of knowledge, skills, and attitudes.

Key words: Programming for Students, Scratch, DigComp.

Introduction

In recent years, numerous initiatives have been undertaken in many countries around the world to encourage children to learn programming. There is a clearly defined educational policy whose aim is not only to train children to work with digital technologies and specialised software from an early age, but also to turn them into creators of software products. In the context of this educational policy in Bulgaria, a special subject (Computer modelling) was created for pupils in grades III and IV [1], and the IT curricula in grades V, VI and VII were supplemented with a part related to the creation of projects in the field of computer modelling and programming. This development is in line with the political aspiration to digitise the economy and society worldwide, including the European Union [2].

In this regard, it is essential to provide a concise review of the history of programming in the classroom and highlight current trends, with a particular focus on students in elementary education stage (in grades 3 and 4).

A Brief History of the Emergence of Programming in Classrooms

The concept of integrating computer programming into classroom instruction dates back to the late 1960s, when Seymour Papert and his colleagues at MIT developed LOGO, the first programming language specifically designed for children. Through this text-based language, students could input commands to control a 'turtle,' which moved and drew geometric shapes by dragging a pen, thereby introducing young learners to computational thinking and problem-solving [3].

The LOGO programming language was specifically designed as an educational tool. Its key features – modularity, extensibility, interactivity, and flexibility – were intentionally developed to support and enhance the learning process [4].

LOGO programming activities encompass various fields, including mathematics, language, music, robotics, telecommunications, and science. This programming language is utilised for developing simulations, creating multimedia presentations, and designing games. It is specifically designed to be accessible to beginners, including young children, while also supporting commands and instructions for the execution of more complex projects by experienced users [4].

Papert firmly adheres to constructivist theory, asserting that the true potential of programming languages emerges when they facilitate the design and creation of personally meaningful, computationally rich projects that encourage children to think in innovative ways [3].

However, it is important to acknowledge that despite the initial enthusiasm for teaching all children how to program (with the advent of personal computers in the 1970s and 1980s.), many schools have subsequently shifted their focus to using computers for other purposes [5].

Although computers have become widely accessible over the past 20 years (including to children in the 6-18 age group), very few children have acquired programming skills. Furthermore, programming is often

perceived as a highly specialised activity suitable only for a small segment of the population [5].

Thus, a question arises: Why has LOGO failed to serve as a motivating force for children to learn programming in this language? Mitchel Resnick, a former student of Papert and the director of the organisation that developed the Scratch programming language [6], offers a plausible explanation for this phenomenon [5].

- Many children encounter difficulties in mastering the syntax of programming languages.
- Introductory programming projects, such as generating lists of prime numbers or creating drawings with lines, often fail to engage young learners and do not align with their interests.
- Programming is frequently introduced in contexts where there is a lack of expertise to provide guidance when challenges arise or to promote deeper exploration when students experience success.

The Scratch Programming Language

In developing LOGO, Papert followed the philosophy that the language should possess characteristics of a "low threshold and no ceiling". This concept aims to make the programming language accessible to beginners, including young children (low threshold), while also enabling more experienced users to undertake complex projects [4].

The developers of Scratch (Scratch is a programming language particularly popular among students in the 1st to 4th grade age group, was developed by the Lifelong Kindergarten research group at the MIT Media Lab.) incorporated these features and introduced an additional characteristic known as "broad sides" [5]. This feature refers to the integration of appropriate tools designed for implementing various types of projects, thereby attracting individuals with diverse interests and learning styles to programming.

Currently, Scratch is the preferred programming language for children [7] and it is utilised by millions of young people worldwide. Additionally, a platform has been established that enables learners to create their own code, exchange ideas, share best practices, and express their opinions in a forum. Scratch functions as a visual programming environment, allowing users to learn computer programming while developing personally

meaningful projects, such as animated stories and games. A major design objective of Scratch is to promote independent learning through experimentation and collaboration with peers [8].

A wide range of projects has been developed using Scratch, including animated stories, games, online news shows, book reports, greeting cards, music videos, science projects, tutorials, simulations, and sensor-driven applications [8]. This extensive variety makes Scratch the largest free programming community for children globally.

Currently, Scratch is incorporated into the curricula of numerous schools worldwide, including those in Bulgaria. The language is designed to be user-friendly, interactive, and engaging, categorising it within the realm of block-oriented programming languages. As such, Scratch serves as an effective tool for teaching programming in many educational systems across various countries.

However, Scratch presumes that children possess foundational skills in reading and writing, as the programming blocks include words that correspond to the actions they execute. Additionally, learners are expected to manage complexity and navigate the virtually limitless possibilities of adding commands [3, 5].

Ultimately, the fundamental philosophy of Scratch emphasises the cultivation of computational thinking [5], utilising the principles of constructivism, which are inherently integrated into the design of the Logo programming language [3].

Computation-based thinking is a problem-solving process characterised by the following elements [9]:

- Formulating problems in a manner that enables the use of computers and other tools to assist in solving them.
- Organising and logically analysing information.
- Representing information through abstractions, such as models and simulations.
- Automating solutions through algorithmic thinking, which involves establishing a sequence of orderly steps to reach a solution.
- Identifying, analysing, and applying potential solutions to achieve the most effective and efficient combination of steps and resources.

- Generalising a method for solving a specific problem in order to apply it to a broader range of issues.

Using Scratch in New Educational Contexts

Drawing upon my teaching experience in programming with students from grade 1 to grade 12, I propose the following approaches for integrating Scratch more actively into elementary education:

- Creating and editing vector and raster images.
- Creating and editing sound files.
- Creating projects using lists.
- Creating projects using various forms of artificial intelligence.

The proposal is essentially about using the programming language in new learning contexts.

However, several unresolved questions persist, particularly regarding the design and implementation of an appropriate curriculum, the engagement of students through stimulating lessons and projects, the effective utilisation of children's cognitive resources to facilitate the acquisition of knowledge, skills, and attitudes, as well as the application of suitable motivational strategies and gamification techniques within the learning process.

All of these topics and challenges are directly pertinent to ensuring high-quality programming education for adolescents.

In addition, it is essential to clearly articulate the significance and role of programming-related curricular topics for students within the framework of developing 21st-century competencies.

Digital Competence Framework for Citizens

The member states of the European Union have identified, as a political priority, the necessity for citizens to recognise that digital competence is a skill that must be cultivated throughout life [10].

In this context, the European Digital Competence Framework for Citizens (DigComp 2.2) provides a common language for identifying and describing key areas of digital competence. It serves as a comprehensive tool in the EU for enhancing citizens' digital competence, assisting policymakers in formulating policies that support the development of digital skills, and planning educational and training initiatives aimed at improving

the digital competence of specific target groups [11].

According to the document, there are 21 digital competencies organised into five areas: information literacy, communication and collaboration, digital content creation, safety, and problem-solving.

Figure 1. Conceptual reference model of DigComp [11]

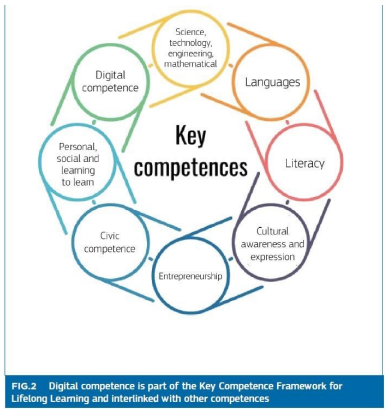

Figure 2. Conceptual reference model of DigComp [11]

DigComp 2.2 provides an integrated framework that includes over 250 examples of digital competencies in EU citizens, encompassing knowledge, skills, and attitudes.

For the purpose of this paper, it is essential to clarify the meaning of Competence 4: Programming within Domain 3: Digital Content Creation, and to provide examples from the document illustrating how this competence is assessed. The programming competence involves skills to "plan and develop a sequence of understandable instructions for a computing system to solve a given problem or to perform a specific task." [11]. Below are some of the most important examples.

Examples of an individual's digital programming competence are provided in the knowledge domain. The individual is expected to know that:

"151. Knows that computer programs are made of instructions, written according to strict rules in a programming language.

154. Knows that programs produce output data depending on input data, and that different inputs usually yield different outputs . . . "

In the area of skills, the individual's capabilities are evaluated based on the following criteria [11]:

"160. Knows how to combine a set of program blocks. . . , in order to

solve a problem.

161. Knows how to detect issues in a sequence of instructions, and make changes to resolve them . . . "

In the area of attitudes, the focus is on personality traits and their relevance to [11]:

„164. Willing to accept that algorithms, and hence programs, may not be perfect in solving the problem that they aim to address.

165. Considers ethics. . . "

Conclusion

Although programming began to be integrated into the curricula of some schools around the world as early as the late 1960s, its widespread adoption occurred after the introduction of the Scratch programming language. Scratch, which uses a block-oriented programming approach, is now the most popular programming language worldwide among primary school students. This language is used in the educational systems of many countries, including Bulgaria, to acquire initial knowledge, skills and attitudes in the field of programming. Creating the prerequisites at an early age for acquiring programming competences is part of the objectives included in the European Digital Competence Framework for Citizens (DigComp 2.2). While Scratch provides rich opportunities for the creation of personally meaningful projects, in a purely educational aspect, ways could be sought to use the platform in new learning contexts, such as the creation and editing of raster and vector images, sound files, the creation of projects using arrays and those using various forms of artificial intelligence.

References

[1] Ministry of education and Science, General Education directorate, Study programme for Computer modelling for 3rd grade (in Bulgarian), retrieved 20.11.2024 from `https://www.mon.bg/nfs/2018/01/up_km_3kl.pdf`.

[2] European Commission, Digital Education Action Plan 2021–2027 Resetting education and training for the digital age, COM (2020) 624 final, 2020, retrieved 20.11.2024 from `https://education.ec.europa.eu/focus-topics/digital-education/action-plan`.

[3] M. Bers, Coding and Computational Thinking in Early Childhood:

The Impact of ScratchJr in Europe, *European Journal of STEM Education*, Vol. 3 (3), 2018, https://doi.org/10.20897/ejsteme/3868.

[4] Logo History, retrieved 20.11.2024 from https://el.media.mit.edu/logo-foundation/what_is_logo/logo_programming.html.

[5] M. Resnick, J. Maloney, A. Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, Y. Kafai, Scratch: Programming for Everyone, *Communications of the ACM*, Vol. 52, No. 11, 2009, 60–67, https://doi.org/10.1145/1592761.1592779.

[6] Mitchel Resnick, retrieved 20.11.2024 from https://www.media.mit.edu/people/mres/overview/.

[7] About Scratch, retrieved 20.11.2024 from https://scratch.mit.edu/about.

[8] J. Maloney, M. Resnick, N. Rusk, B. Silverman, E. Eastmond, The Scratch Programming Language and Environment, *ACM Transactions on Computing Education (TOCE)*, Vol. 10, Issue 4, Article No.: 16, 2010, 1–15, https://doi.org/10.1145/1868358.1868363.

[9] P. Plaza, M. Castro, J. Sáez-López, E. Sancristobal, R. Gil, A. Menacho, Promoting Computational Thinking through Visual Block Programming Tools, *2021 IEEE Global Engineering Education Conference (EDUCON)*, Vienna, Austria, 2021, pp. 1131—1136, DOI: 10.1109/EDUCON46332.2021.9453903.

[10] European Commission, Eurydice Report, 2019, Digital Education at School in Europe, p. 4, retrieved 20.11.2024 from https://eurydice.eacea.ec.europa.eu/publications/digital-education-school-europe, ISBN: 978-92-9492-994-5, DOI: 10.2797/763.

[11] European commission, DigComp 2.2 The Digital Competence Framework for Citizens, 2022, p. 2, retrieved 20.11.2024 from https://op.europa.eu/en/publication-detail/-/publication/50c53c01-abeb-11ec-83e1-01aa75ed71a1/language-en, ISBN: 978-92-76-48882-8, DOI: 10.2760/115376.

Georgi Hristov,
Sofia University "St. Kliment Ohridski",
Faculty of Mathematics and Informatics,
5 James Bourchier Blvd., 1164 Sofia, Bulgaria
Corresponding author: gphristov@uni-sofia.bg