A GLANCE BACKWARD WITH NOSTALGY AND FORWARD WITH OPTIMISM
(OR DO WE NEED TO START FROM SCRATCH WHEN INTRODUCING CHILDREN TO PROGRAMMING)

Evgenia Sendova[1], Roumen Nikolov[2], Pavel Boytchev[3]

[1] Institute of Mathematics and Informatics at the Bulgarian Academy of Sciences
[2] University of Library Studies and Information Technologies (UNIBIT)
[3] Faculty of Mathematics and Informatics, Sofia University
jenny@math.bas.bg, roumen.v.nikolov@gmail.com, boytchev@fmi.uni-sofia.bg

A language that doesn't affect the way you think about programming is not worth knowing.
Alan J. Perlis

***Summary**: The authors represent three generations of informatics educators who profess the Logo educational philosophy, promoting the programming as a means for active learning, formulating of materialized hypotheses and creative self-expression.*

***Key words**: Introduction of programming to children, Logo, Logo versions, Scratch*

1. Impressions of certain contemporary views on using IT in education

At an international exhibition marketing innovations in education technology attended by more than 40 thousand visitors there were a great number of examples of how NOT to use technology. A program generating a sentence of randomly chosen words was called "Poet". Similarly, a program generating a sequence of randomly chosen notes was called "Composer". The authors had not implemented any concepts related to the structure of a poem or a musical composition. Not surprisingly, they had not heard of books such as "Exploring Language with Logo" [1].

At a very recent issue of an annual national conference embracing researchers, educators and teachers in mathematics and informatics education there was a workshop led by young representatives of companies promoting programming courses for young children. They were demonstrating a couple of robots moving in a maze on the floor. After showing the commands needed to move the robots (not LEFT, RIGHT and FORWARD as you might expect) a mother from the audience asked: *What would you do after a child who has already learned these commands gets bored?* The answer was: *No, problem at all! We have five other programming environments to offer…*

Such examples make us feel nostalgic about what could be referred to as "oldie but a goldie".

2. A Glance Backward – the RSI experiment

In the early 80's a model for an ICT-prompted curriculum was designed and launched by a Research Group on Education (RGE) as a joint project of the Bulgarian Academy of Sciences and the Ministry of Education. The RGE mission was to develop a novel curriculum based on the use of computers as one of its natural components and means for integration of school subjects [2]. The educational resources developed specially for the experimental 29 schools included textbooks, teacher guide-books, a bulletin "Informatics and mathematics for teachers", and unified (Logo-based) computer microworlds. These microworlds were tuned to specific subject domains, the main idea being to encourage children to act as scientists rather than learn about science [3, 4].

During the first four years, informatics was introduced as a part of an encyclopaedic education. One of the main integrated disciplines in the primary cycle was "I read, write and calculate". A good example of activities within this subject would be creating a situation, in which children would decode a letter (matching numbers with letters), read, write, and code a return message in a context of interest to them.

An innovative idea of integrating the study of mathematics, natural languages (Bulgarian, English and Russian), and a computer language (Logo in this case) was launched in 1984 by creating the textbook Language and Mathematics for 5th and 6th grade (corresponding to the first and second grade according to the RGE system, Fig. 1).

Fig. 1 Experimental textbooks in the Language and Mathematics subject

Designed to show the intersection of language study with mathematical thinking in the context of informatics, these experimental textbooks included problems on translating from a natural to a formal (scientific) language, algorithmic description of basic grammar rules, and ways to extend the Logo vocabulary to several languages. Informatics notions, such as *coding, decoding, tree-graphs, algorithms, variables,*

tables, procedures, recursion, data, etc. were applied in the context of playing, editing and creating educational games, coding and decoding secret texts, describing and executing algorithms in mathematics, creating models of language, music and science phenomena.

Here is what Paul Goldenberg (invited as an expert in mathematics and informatics education) wrote about the RGE approach [5]:

One's own natural language is best for conveying the semantics of a mathematical idea or situation; algebraic language is best at expressing and transforming quantitative or structural relationships; and computational language is optimal for describing processes and algorithms. That – especially the last two paragraphs – I feel like I learned in Sofia!

Specifically designed microworlds provided tools for the students to deal with new notions from a procedural rather than from a declarative point of view. This has already had an impact on the way we started teaching mathematics, literature, art and music. The newly developed curriculum enabled students to pass gradually from constructing controllable models, through turtle geometry and other problem-oriented microworlds to a fully programmable microworld for explorations in Euclidean geometry – *Geomland* [6]. The Geomland microworld was launched in 1986 as a language based computer laboratory enabling pupils to construct and experiment with Euclidean objects, to investigate their properties, to formulate and verify conjectures, i.e. to do and discover mathematics [7].

The authors' experience (which might be called *experience of three generations of programmers*) shows that the duality of Logo as a programming language and environment, on one hand, and educational philosophy and culture, on the other, is one of its biggest strengths.

3. How many versions of Logo are there?

Although the use of Logo language declines in the past decade, the Logo philosophy (known also as *constructionism* [8, 9, and 10]) is still well alive and it reincarnates itself in numerous learning environments and educational contexts. The tree of Logo implementations (all the different Logo versions and dialects build in the past few decades) could be seen in Fig. 2 [11].

A fundamental feature of the first-born Logos is the concept of the turtle that can be animated by students via programs. The first turtles were robots, crawling on the floor and leaving traces into beautiful geometrical patterns. These turtles were real physical objects that students can touch and can associate with. This tangibility was one of the features of Logo that made it stand out compared to all then-existing educational languages.

The physical turtle was soon replaced by a small triangle on the computer screen. This made Logo turtles much more affordable – no need to buy a robot, no need to

replace the pen, no need to repair the motor. Unfortunately, this improvement was at the cost of losing tangibility. The new turtles could be touched only imaginarily. As new educational environments emerged, they improved the functionality of the virtual turtle (like 3D motion, nice visual appearance, multiple instances, faster and more complex behavior), but still it remained effectively intangible.

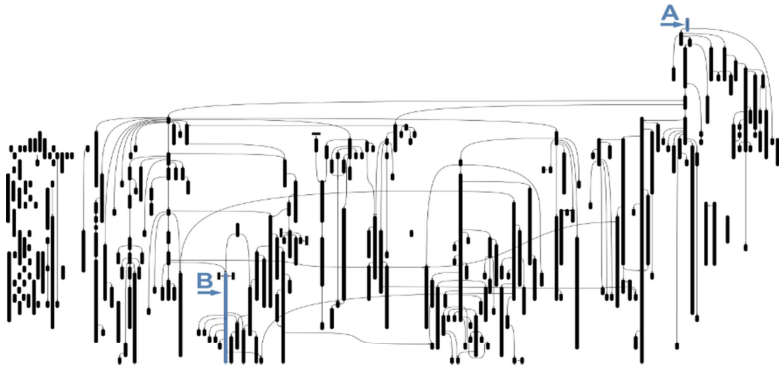


Fig.2. The Logo tree: in the top-right corner, marked by "A", is the original Logo. The most wide-spread successor of Logo and the Logo philosophy, Scratch, is marked by "B" (in the middle of the left section)

4. Do we need to start from Scratch today?

The pun is intended. Indeed, *Logo* is not a fashionable word anymore. People's reactions to it are either *yeah-I-used-it- when-I-was-a-kid* nostalgia or *hum-what-is-this-supposed-to-mean* confusion. However, most of the features first demonstrated with Logo are found in modern educational environments.

One of the many new block-based programming environments developed specifically for young users is Scratch. It has over 2 million registered users aged less than 12 years. The popularity of this style of programming with young children is in part due to *its ease of readability, browsability, interactivity and visual and dynamic outcomes* [12].

Let us note that in Scratch the code blocks only lock into place in syntactically valid ways, thus "bugs" are always semantic errors, unlike the text-based Logo versions, where the "bugs" could be due to typing errors or misremembered details of language syntax. One would expect that the number of affordances offered by Scratch would make it easier for students to learn and interpret.

However, recent study [13] comparing the attitudinal and learning outcomes of sixth graders programming in either Logo or Scratch shows that students who learned Logo had on average higher confidence in their ability to program, contrary to researchers' hypothesis. The authors of the study suppose that the low-level details

(such as syntax) in the textual representation of Logo allow students to focus on some important low-level details, such as the role of every command in a larger program. Furthermore, they conjecture that it is this low-level focus which allowed students learning Logo to compensate for the lack of a visual representation.

By comparing the performance of students in different programming environment one could decide which features of the student experience are influenced by the content of the programming, and which – by the programming environment.

But maybe it is true that *if a computer language does not change your way of thinking there is no need to learn it*…

Here is how the idea of modelling a fern has been implemented in the Bulgarian version of Comenius Logo (Fig. 3 left) and in Scratch (Fig. 3, right).

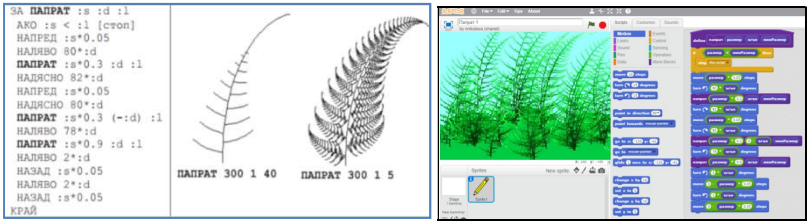


Fig 3. Modeling ferns in Comenius Logo (left) and Scratch (right)

An iconoclastic implementation of the Logo language was *Elica*, designed and developed by the third author [14]. *Elica* reached far beyond the traditional flat turtle graphics by providing a 3D animation of user-programmable objects.

Elica was used to teach Computer Graphics at Sofia University [15]. Next we give examples taken from two other courses at the Faculty of Mathematics and Informatics in which the objects are defined in a Logo style.

The snapshots on Fig. 4 are of a model of the human figure. It is used in the *Geometry of Motion* course where students use virtual mechanical elements to construct programmatically a complex mechanism. Internally the linkage between the elements is implemented by an invisible turtle, which traverses the "joints" and the "bones" similarly to how it would traverse a Pythagorean three. The examples could be played live at this URL: https://goo.gl/j9yGhG.

The snapshots on Fig. 5 are from *Fundamentals of Computer Graphics* course. They use a Logo-like set of commands to define and visualize: dynamically-random Pythagoras tree (left), animated hand gestures (middle) and a flexible skin over a skeleton (right). These examples can be played live at https://goo.gl/dLDM6n, https://goo.gl/YKmFVj and https://goo.gl/cuVKjt.

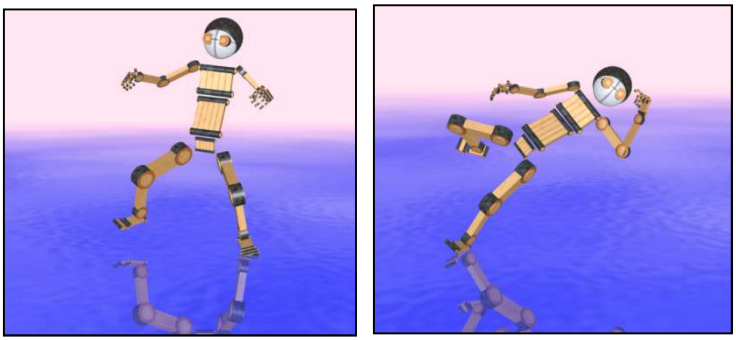


Fig. 4 A model of the human figure used in the Geometry of Motion *course*

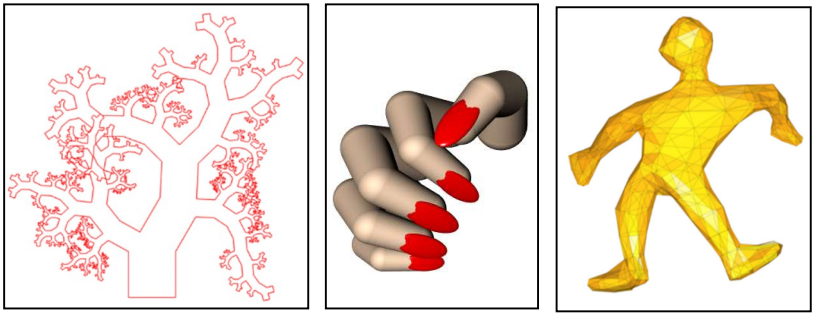
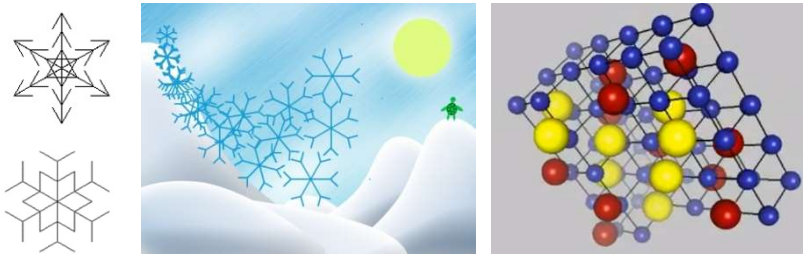


Fig. 5 Dynamically-random Pythagoras tree (left), animated hand gestures (middle) and a flexible skin over a skeleton (right).

Models of phenomena from natural sciences have been often part of projects developed by young students and teachers-to-be (Fig. 5 and 6).



(a) (b) (c)

Fig 6. (a) Snowflakes with Comenius Logo; (b) Winter whirlwind with Scratch; (c) Atoms forming a crystal structure with Elica

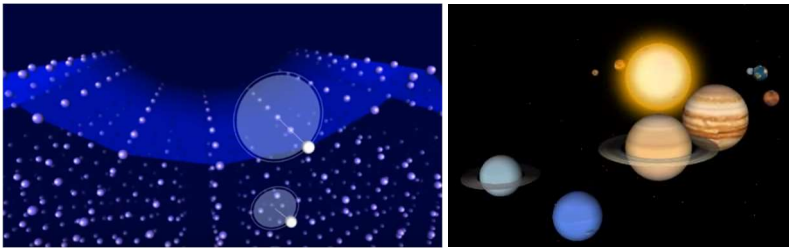


Fig. 7 Modeling wind generated waves (left) and the Solar system including Pluto (right)

Conclusion

A French proverb says: *Plus ça change, plus c'est la même chose*. A translation in English offered by Douglas Hofstadter in his wonderful collection of articles on language, art and logic *Metamagical Themas* [16] reads as follows: *The more it changes, the samer it gets*. The idea is that the more manifestations of a certain phenomenon you see, the more you are able to see what the sameness is and thus - to understand its essence [17].

This idea was well demonstrated at a recent event (14th December 2017) at the Institute of Education of University College London celebrating the work and contributions of two eminent representatives of Constructionism (Celia Hoyles and Richard Noss) who are Foreign Fellows of the Union of Bulgarian Mathematicians (Fig. 8 left). They were editors of "Learning mathematics and Logo" [18], they coordinated the *WebLabs* project, which focused on using the *ToonTalk* computer environment for young students to build models of their mathematical and scientific knowledge [19], and are recently working on the *ScratchMaths* project, which has designed curriculum materials and professional development to support mathematical learning through programming for pupils aged between 9 and 11 years [20].

Fig. 8 left: Richard Noss and Celia Hoyles share their memories with colleagues, students and academic friends from around the world; right: Paul Goldenberg talks about language and mathematics (quoting the Bulgarian RSI experiment)

At the same event the above mentioned Paul Goldenberg talked about *his fascination by language and how kids learn and use it.* He explicitly mentioned his being inspired by both the recent project *ScratchMaths* and the Bulgarian experiment on integrating *Language and Mathematics* within the Research Group on Education (Fig.8 right).

Thus, there is no need to put in conflict the different programming environments provided they bring to life an inspiring educational idea – *it is better to program to learn rather than to learn to program*.

The recent advances in computer technology, especially in the area of Augmented Reality (AR), are a chance to bring back the direct contact between students and the Logo turtle … and to go back to the roots of Logo. It is already possible to build educational environments where students could directly manipulate AR objects. It is a matter of time to have this technology widely spread in schools. The real question is "What shall the students do with what they know and can…" The answer is a joint responsibility of the technology designers, the teacher educators and the teachers.

References

1. Goldneberg, E. P. & Feurzeig, W. *Exploring Language with Logo,* The MIT Press, Cambridge, MA, 1987
2. Sendov, Bl. Education for an Information Age, in *Impact of Science on Society*, v37, n2, 1987, pp.193-201
3. Nikolov, R. Integrating Informatics into the Curriculum, in *Education & Computing*, North-Holland, 1987, pp. 369-374
4. Sendova, E. Assisting the art of discovery at school age – a Bulgarian experience, Third chapter of *Talent Development Around the World*, coordinator Pedro Sánchez-Escobedo, Mérida, Yucatán, 2013, pp. 39-98
5. Goldenberg, E. P. Bringing Back Formal Language: A Use to Counter My Worries about Computers in the Mathematics Classroom. In: Proceedings of the Seventh European Logo Conference Eurologo'99 (Eds R. Nikolov, E. Sendova, I. Nikolova, I. Derzhanski), Sofia, Bulgaria, 22–25 August, 1999.
6. Sendov, B., Dicheva, D. A Mathematical Laboratory in Logo Style, in Lovis, F. and Tagg E.D. (Eds.), *Computers in Education* – Proceedings of the ECCE'88, Lausanne, Switzerland, North Holland, 1988, p. 213
7. Sendova, E. Enhancing the Scientist into the Pupil: a Computer Environment Supporting Discoveries in the Classroom, in Aiken, R. (Ed.) *Education and Society*, Information Processing 92, vol. 2, Elsevier Science Publishers B. V. (North Holland), IFIP, 1992
8. Papert, S. What is Logo? Who needs it? In *Logo Philosophy and Implementation*, LCSI, 1999
9. Sendova, E. You do – you understand, you explore – you invent: the fourth level of the inquiry-based learning, in Futschek, G., Kynigos, C. (Eds.) *Constructionism and Creativity*, Proceedings of the 3d International Constructionism Conference, August 19-23, Vienna, Austria, pp. 103 – 112
10. Sendova, E. Constructionism as an Educational Philopsophy and a Culture - A Tribute to Seymour Papert (in Bulgarian), in *Mathematics and Education in Mathematics*, 2017

Proceedings of the Forty-sixth Spring Conference of the Union of Bulgarian Mathematicians Borovets, April 9–13, 2017, pp. 29-51
11. Boytchev, P., Logo Tree Project, 2014, http://www.elica.net/download/papers/logotreeproject.pdf
12. Weintrop, D. & Wilensky, U. To block or not to block, that is the question: students' perceptions of block-based programming. In Proc. IDC'15. ACM, 2015, pp. 199-208
13. Lewis, C. M. How programming environment shapes perception, learning and goals: Logo vs. Scratch. In SIGCSE '10 Proceedings of the 41st ACM technical symposium on Computer science education, 2010, pp. 346-350
14. Boytchev, P. *Elica*, http://www.elica.net/
15. Boytchev, P., Design and Implementation of a Logo-based Computer Graphics Course, in *Informatics in Education*, Volume 6, No. 2, 2007, Vilnius: Institute of Mathematics and Informatics, pp. 269-282
16. Hofstadter, D. *Metamagical Themas: Questing for the Essence of Mind and Pattern,* 1985, Basic Books, ISBN 0-465-04566-9
17. Gachev, G., Sendova, I. Nikolova, E. The-more-it-changes-the-samer-it-gets Principle in the Context of Mathematics and Informatics Education. In Gregorczyk, G. et al. (Eds.) Proceedings, EUROLOGO 2005, pp. 87-99. ISBN 83-917700-8-7
18. Hoyles, C., Noss, R. (eds.) Learning mathematics and Logo, 1992, MIT Press
19. Kahn, K., Sendova, E, Sacristan, A.I., Noss, R. Young Students Exploring Cardinality by Constructing Infinite Processes, *Technology, Knowledge and Learning*, 2011, DOI: http://dx.doi.org/10.1007/s10758-011-9175-0
20. Benton, L., Hoyles, C., Kalas, I. et al. Bridging Primary Programming and Mathematics: Some Findings of Design Research in England, *Digit Exp Math Educ*, 3: 115, 2017, https://doi.org/10.1007/s40751-017-0028-x

ПОГЛЕД НАЗАД С НОСТАЛГИЯ И НАПРЕД – С ОПТИМИЗЪМ

Евгения Сендова, Румен Николов, Павел Бойчев

***Резюме**: Авторите са представители на три поколения програмисти и преподаватели по информатика, за които Лого е не само език и среда за програмиране, а най-вече образователна философия и култура, известна под името „конструкционизъм". В Лого програмирането се разглежда като средство за активно учене, формулиране и верифициране на материализирани хипотези, както и за творческо себеизразяване. В статията са представени първите образователни ресурси за програмиране с Лого в рамките на Проблемната група по образованието (1978-1990). Наличието на огромен брой имплементации (версии и диалекти) на Лого, създадени през последните няколко десетилетия, е илюстрирано с дърво, в което се открояват оригиналната версия на Лого и най-разпространената съвременна негова версия – Scratch. Направен е кратък коментар върху резултатите от изследвания дали съществува съществена разлика в поведението на ученици при работа с класическо Лого и Scratch. Разгледани са и примери от два курса във ФМИ-СУ, в които се ползва лого-стил на дефиниране на обекти. Изразена е (основателната) надежда на авторите близкото бъдеще на Лого да се върне към неговите корени.*