A step back into the future

Recovering Papert's lesson using free software tools: text-based coding as a cross curricular teaching and learning tool to foster creative thinking skills in primary and secondary classrooms under the European cooperative umbrella action of eTwinning

Andreas R. Formiconi & Francesca Mancini

arf@unifi.it

7th June 2017

University of Florence

Abstract

This draft is to share a vision about coding and computational thinking at school seeking confrontation and, possibly, collaboration with other TTI members. We begin by putting in context the text-based versus block-based coding issue. Then we try to make a step back to the basic thoughts of Seymour Papert about how using computers to develop new skills and to acquire new knowledge entails cognitive aspects but feelings as well. The concept of free software is presented and how it resonates with the European Funding values, especially within educational contexts. Among the free software applications, we focus on the Office suite LibreOffice with emphasis on the multilingual features and the LibreLogo functions, an implementation of Seymour Papert's Logo language which allows to build and embed vector graphic images in text documents. Finally, the role of our university and primary schoolteacher curriculum are described.

Keywords: eTwinning, computational thinking, coding, Seymour Papert, Logo, LibreLogo, free software

Introduction

In the "Growing Digital Citizens" eTwinning publication digital citizenship is described as being based on three pillars: belonging, engagement and protection (Ferrari & Martens, 2016, p. 11). The respective senses of belonging, engagement and responsibility, are all intertwined but they can be fully developed only by means of appropriate digital skills, nowadays. However, what do we actually mean by "digital skills"? The question is relevant since they may involve quite a broad range of competencies, but not all of them require the same degree of commitment as well as the same cognitive involvement. The attribute of “digital native” is by no way sufficient to grant, in a thorough and productive way, the senses of belonging, engagement and responsibility. A large majority of youth is just familiar with digital environments but here the term digital is somewhat misleading. For instance, the ability to sign up to some online service, or the ability to move around the buttons of whatever interface, is just a matter of habit, but such skills are scarcely related to any relevant competencies, usually they are not. The awareness about the digital nature of data and the way things happen in computers or networks is limited. This means that what people can do is mostly determined by features of graphic user interfaces, which depend upon software design and commercial strategies, and not so much on mastery of the universal symbolic systems underpinning natural languages and mathematical thought. It is by means of manipulation of those systems of symbols that one is able to develop deep thought and thorough comprehension of facts. The practice of coding, in its traditional form, involves the manipulation of symbolic information and it is generally recognized that it allows the students to create and not only to use. Actually, the discourse on coding practices is embedded in the broader one on computational thinking. Recently, the EC has devoted an extended study to the state of the art of computational thinking (Bocconi, 2016). What it comes out is that if the sense of urgency to introduce computational thinking in compulsory education is quite strong, at the same time a lack of consensus is still there, even about the mere definition of the concept. Thus, we find ourselves in the uncomfortable situation of acting urgently but without having the possibility of being in control. Things move fast, actually. The time to give shape to the significant lessons learned through the ongoing experiences and to take fully advantage of them lacks. This is exactly what is happening to the evolution of coding practices in the educational context. To tell it in few words, it seems that, in front of the explosion of languages and devices of any kind, the sound, deep pedagogical and technical motivations that inspired the earlier educational coding experiences have vanished in a sparkly cloud of fancy activities.

Nowadays in most school contexts coding is synonymous of Scratch or, at any rate, of blocks-based languages. These languages are quite smart for providing a first programming experience to kids. And they are even powerful, allowing for a broad range of coding experiences. Moreover, Scratch fostered the spread of coding a great deal, through its social platform. However, within the whole process, the general idea of coding leaned over the production of animations, which might be fine, because to realize them some quite advanced programming methods are required. What are we missing then? To understand this we have to recall the lesson of Seymour Papert.

A step back into the future

The idea of including computer programming among the educational activities is due to Seymour Papert. Papert, a South African mathematician, arrived in the United States in the mid 1960s after having worked with Jean Piaget for five years. He released the first version of Logo in 1967, when working at the MIT Artificial Intelligence Laboratory. Logo was an advanced language conceived at the intersection between the fields of artificial intelligence and developmental psychology, as a tool for improving the way children learn and solve problems. It's key idea, using Papert's famous expression, was to allow for a *low floor and a high ceiling*. For this reason, even if apparently simple in the first steps, its inner architecture allowed users to extend their capabilities in a virtually limitless fashion. A great number of educational languages have been derived from Logo, among which Scratch, by far the most successful. Scratch is a relative of Logo being developed by Mitchel Resnick, a former student and successively coworker of Papert at the MIT. Actually, the basic functionalities of Logo can also be found in Scratch that, however, has many more features, among which the blocks-based instead text-based interface and the possibility to build animations or true video games. On the other side, Logo was thought as a way to explore mathematical concepts in a *body-syntonic* way, another papertian expression which refers to the idea of building a geometry - the Turtle Geometry - in analogy with the body geometry which is well known by kids, before they get in touch with formal math. So what, if Logo is in some way included in Scratch? The fact is that all the considerations on what is actually going on when kids are let exploring with Logo, all the awareness about the importance of personal discovery of mathematical concepts, all the strong emphasis on creative approach to study scientific ideas have almost completely disappeared. Not because Scratch, or other similar languages, makes such perspectives impossible but because the whole interface is too much skewed towards the childish side. This does not mean that you cannot do quite complex stuff with Scratch, even extremely complex ones. Instead, it's about the fact that most of the activities done in Scratch are about the production of animations and simple games, and by far most of the projects are very basic and short lasting, as it has been shown by some recent studies based on scraping of the Scratch database (Aivaloglou & Hermans, 2016; Matias, Dasgupta & Hill, 2016; Scaffidi & Chambers, 2016).

Ironically, the childish flavor, thought to facilitate the introduction to programming, turns out to be a limiting factor, basically because students crave hard: if you complete a hard task you have proven yourself, if you fail... after all it was not so easy (Krouse, 2016A). Paradoxically, Scratch may be frustrating because everything seems so easy but soon it might get much harder. Because coding it's hard. Like math. Making life much easier is not always a good idea. A number of studies revealed that a Scratch introduction to programming does not necessarily facilitate the transition to conventional coding languages (Lewis, 2010; Lewis, Esper, Bhattacharyya, Fa-Kaji, Dominguez & Schlesinger 2014; Weintrop & Wilensky, 2015). That's why new approaches for easing the transition to "true languages" are emerging (Homer & Noble, 2014; Price & Barnes, 2015; Krouse, 2016B).

But still something is missing. Let's recall some reasoning of Seymour Papert (Papert, 1986):

> *Welcome to the Logo tapes. These tapes are about logo but they are not just about logo, beyond logo they are about thinking. They are about how to think about computers, and how to use computers to think about other things. They are about how to use a Logo experience, to develop new thinking skills for yourself as much for you students. But even beyond thinking, the tape have much to say about feelings. People, adults as well as children, have strong feelings about computers, and their experience with computers influence the way they feel about many other things. For example, about school, about learning and most relevantly here, people experience with computers often influences the way they think about themselves.*

The lesson of Papert is by no way a purely technical one. Nor it's limited to specific competencies, or accessing information, sharing and so on. Papert's idea of using computers in education is a holistic one. Logo was conceived to explore geometry, math, or even science, by means of clever simulations. But even more than that:

> *The main purpose of Logo is not what they call "computer literacy" - of course it serves that, based on anything else I can think of, but the real purpose is not to have better understanding of computers but through computers to have better understanding of everything else including, I'd like to say, yourself.* [...] *I'm not trying to give you a theory*

of what causes children to be so involved and engaged with a computer, I'm trying to encourage a way of thinking that looks beyond the role of the computer in teaching one or another corner of the curriculum and tries to look at the emotional roots of what's going on.

Papert's thought emphasises the pleasure and benefit of discovering learning, appropriation, making knowledge your own in a way you feel good about it, seeking resonance between the immediate learning experience and the larger experience that makes up the learner's life.

In this tape, I tried to show how a teacher can use Logo to play the role of intellectual glue, the role that mathematics has made for me. At other end, by some reflections, unpacking the intuition every teacher has, that is good to make connections... well why? There is a cognitive side: connections help you understand, you understand the new by referring to the old, they help you remember. But there is a deeper side, one that has to do with how you feel about knowledge and how you feel about yourself. Connecting new knowledge to things you know and love and things you can do makes you feel good about it, makes you take it in a form that is your own, but taking knowledge in form that feels to you as you, you change your feelings about you as well. You no longer think about yourself as somebody who can do math but doesn't really understand poetry, or can draw but doesn't have the head for numbers. Instead, you appropriate all knowledge in a form that is yours, that you can do, that you can love. And through loving what you know you love yourself more.

These words have been extracted from a video series made in 1986. However, even if the technologies used by Papert in these videos may appear quite obsolete nowadays, and even if the software derived from Logo in these thirty years are extremely valuable, we feel that the vision of Seymour Papert still belongs to the future and it is something we still have to strive for. We believe that these considerations give the right tone to bring people from the lower to the upper ladders of the digital participation process described in "Growing Digital Citizens": from watching to sharing, to creating and, finally, to harness the potential of technology for a better society (Ferrari & Martens, 2016, p. 12).

Putting the right pieces together

Free software

According to Article 2, first clause, of the Treaty of European Union:

The Union is founded on the values of respect for human dignity, freedom, democracy, equality, the rule of law and respect for human rights, including the rights of persons belonging to minorities.

According to the Free Software Foundation[1]:

“Free software” means software that respects users' freedom and community. Roughly, it means that the users have the freedom to run, copy, distribute, study, change and improve the software.

[...]

A program is free software if the program's users have the four essential freedoms:

1. *The freedom to run the program as you wish, for any purpose (freedom 0).*
2. *The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for*

[1] Free Software Foundation: https://www.gnu.org/philosophy/free-sw.en.html

this.

3. *The freedom to redistribute copies so you can help your neighbor (freedom 2).*
4. *The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.*

Free software can be adapted by local communities and minorities to suite their specific needs and languages, particularly in the multicultural and multilingual context of the European Union. It is ethic and useful to use it and spread it freely. It counteracts the tendency of breaking proprietary software, which is against the law. It can be modified and improved by anyone who is capable of doing it – and many young people are perfectly able to do it. It fosters collaboration and cooperation on complex shared projects. Free software is a powerful incentive to a creative and ethic approach to the use of technology. It is therefore a relevant instrument of democracy, particularly in educational contexts, in harmony with the funding values of the European Union.

One can use free software by adopting different levels of commitment. The most radical choice is to use the Linux operating system. Linux is a smart operating system with several advantages for most users and nowadays can be installed rather easily. However, even if virtually any user could afford the transition, in practice many users might have reasons to keep their systems, no matter if Windows or Mac OS X. However, free software can be adopted at the much easier level of single applications. There are very good applications which can be installed on all operating systems, such as the office suite LibreOffice[2], the Gimp[3] image editor and the Audacity[4] audio editor, just to mention some among the most popular ones.

[2] LibreOffice is a community-driven and developed software from the not-for-profit organization, The Document Foundation: http://libreoffice.org

[3] Gimp: http://gimp.org

[4] Audacity: http://www.audacityteam.org/

LibreOffice and LibreLogo

The appropriateness of the free software model for multicultural contexts stands out in the case of LibreOffice: some 50 worldwide communities are active to develop and maintain their respective localized versions of LibreOffice[5] but right now 178 languages are supported in some degree, that may include localized user interface, localized help system, auto-text lists, auto-correct list, spell-check dictionaries, hyphenation patterns, Grammar check and Thesaurus (synonyms)[6]. LibreOffice includes all the typical applications of office suites for writing, presenting, organizing data, drawing and so on. But the reason why we are stressing here the interest in LibreOffice is because, among the numerous functionalities there is LibreLogo, a pretty thorough version of the Logo language, available by default among the standard LibreOffice tools since the 4.2.3.3 version (2014).

Recalling Seymour Papert's principle of a low floor and a high ceiling , LibreLogo[7] is a very clever implementation of Logo. The "floor" is extremely low since to begin with you have to enter Writer (the standard LibreOffice word processor), then write down some Logo instructions and run the code just by pushing a menu button. If the code is correct an image is embedded in the document as a standard LibreOffice vector graphics. That way it is extremely simple to begin experimenting with Papert's "Turtle geometry". As we have said, actually Scratch was derived from Logo but, instead of being coded by means of text instructions, it uses colored blocks which can be put together in a Lego-like manner to compose a program. The advantage of this system is that of avoiding the possibility of orthographic and syntactic errors. This may lower the floor at the beginning but successively, it may even hamper the transition to

[5] Native-Lang LibreOffice Projects: http://www.libreoffice.org/community/nlc/

[6] LibreOffice language support: https://wiki.documentfoundation.org/Language_support_of_LibreOffice

[7] LibreLogo: http://librelogo.org

“true languages”, as we have pointed out before. In LibreLogo you have to type text instructions, which at the beginning it may be more demanding but not more than writing English simple sentences. Indeed, it is good that the same kind of skills may be useful in different areas. LibreLogo can be used off line, without having to be connected to a web service, something that can cause some digital divide problems – in many regions this is still an issue. Even the sharing of programs, for exchanging problems and solutions, is extremely easy since it simply requires to send short pieces of text, by whatever means, again without having to rely on an online platform. Finally, with LibreLogo the emphasis is naturally put on math and science, again, which is a good thing since the spread of a true scientific culture is still an issue.

Last but not least, perfectly in the spirit of free software, we got in contact with Németh László, the Hungarian computer scientist who wrote LibreLogo, in order to collaborate to improve the software, following the experience we made in the university courses. We hope to profit from this contact because the objective to foster the development of a relevant European competence on the subject is quite interesting.

The experience in the primary schoolteacher curriculum at the University of Florence

At the University of Florence we made an extensive experimentation of LibreLogo in the Educational Technologies Lab of the Primary Schoolteacher major[8]. The class was composed by 250 students. Moreover, we proposed the same approach to a class of 34 teachers in an online continuous training course. In these classes a text written by one of the authors was used to let the students explore LibreLogo according to the Papert’s Turtle Geometry (Formiconi, 2016).

The basic idea was to foster learning by discovery as much as possible, exactly in the same way the schoolteachers will be expected to do with their pupils. A fundamental role was played by the forum, where the students were encouraged to share problems and solutions,

[8] We are using here the *term* major referring to the Italian *corso di laurea*.

simply by exchanging relevant pieces of text codes within the forum posts. During the 9 weeks of course they wrote more than 400 posts of this kind. Many of them experimented what they learned right in their training activities, whereas the schoolteachers attending the continuous training course brought their newly Logo expertise in their classes and reported feedbacks in the forum. The discovering learning approach was appreciated pretty much, as several students commented: - It seems you are treating us like kids: this is useful for us!

A great deal of ideas and unexpected approaches emerged from the class, with powerful emulation effects. We had those exploring the   drawing fancy alphabetical letters, those who realized digital Tangram figures, those creating a zoo of funny animals, the more math inclined explored the construction of complex geometrical shape and, most interestingly, those that mixed the mathematical control in drawing figures with a kind of aesthetics research, looking, at the very end, for the most pleasant results: kind of STEM to STEAM path.

TTI

The University of Florence is one of the TTI’s of eTwinning.

The class of Primary Schoolteachers is usually composed by 250-350 students. We are already involving the students of this class in eTwinning activities but this is not sufficient, since it takes place at the fifth and last year of the major. We are trying to include an eTwinning training path along the last four years – the major last five years. This requires various organizational tweaks but we are confident we will succeed, hopefully with the next courses this autumn.

After all that we have said, it seems quite obvious to seek a confrontation with other TTIs on the philosophy of this approach about coding at school and, eventually, cooperations on further experimentation and development.

eTwinning methodology strongly contributes to the enhancement of the quality of the teaching/learning process:

- Through a perfect blending of creativity and higher-order thinking skills, it promotes school collaboration in Europe, thus engaging teachers and pupils/students in challenging and non-routinary tasks.
- Through the use of Information and Communication Technologies (ICT) by providing support, tools and services for schools it has always fostered the growth of the educational technology landscape.

The eTwinning TTPilot project 2014-2017 has successfully put a great emphasis on promoting strong bonds among European TTIs and on getting future teachers become acquainted with eTwinning before they arrive in the classroom.

Many are the issues which may concern this eTwinning action: one of the most important and most explored is the growth of future citizens, able to manage the digital transformation of our society marked by more and more intelligent, digital and mesh technologies.

The philosophy of the approach about coding at school actually suggested in this paper might result in the development of digital skills oriented projects where text-based coding practice is integrated in the STEM syllabus at all levels, followed by an active comparison with different European Teacher Training Institutions operating in this field.

Who we are

Andreas Formiconi

I'm in charge as

- "delegato del Rettore" of the university for the development of educational technologies
- referent of eTwinning for the Department of Education.

My background is in Physics – mathematics - computer science and I'm with the Department of Statistics, Computing and Applications.

I'm teaching two classes at the The Department of Education and Psychology:

- Educational technologies in the major of Primary Schoolteacher
- Training processes in the major of Adult Education.

Francesca Mancini

- eTwinning ambassador
- Teacher Trainer for English and ICT
- University Lecturer of TEYL (Teaching English to Young Learners)
- Erasmus Plus Ka1 – Ka2 evaluator for INDIRE
- eTwinning Ambassador
- Among ongoing eTwinning projects STEM: Leading The future[9]

[9] At https://live.etwinning.net/projects/project/119242

References

Aivaloglou E., Hermans F. (2016). *Do code smells hamper novice programming.* IEEE 24th International Conference on Program Comprehension (ICPC).

Bocconi S., Chioccariello A., Dettori G., Ferrari A. and Engelhardt K. (2016). *Developing computational thinking in compulsory education.* ED. P. Kampylis e Y. Punie Science for Policy report by the Joint Research Centre (JRC), the European Commission’s science and knowledge service.

Ferrari A. and Martens H. (2016). Overview on digital citizenship. In Cassells D. et al. (Ed.) *Growing Digital Citizens*. Brussels: European Schoolnet.

Formiconi A.R. (2016). *Piccolo Manuale di LibreLogo*. At http://iamarf.ch/unifi/Piccolo-manuale-LibreLogo.pdf (PDF of version 0.4, 2.6 MB). In Italian, an English version will follow.

Homer M. & Noble J. (2014). *Combining tiled and textual views of code.* Proceeding VISSOFT '14 Proceedings of the 2014 Second IEEE Working Conference on Software Visualization, 1-10.

Krouse S. (2016A). *Scratch has a marketing problem.* At https://medium.freecodecamp.com/scratch-has-a-marketing-problem-f84626bd18ef

Krouse S. (2016B). *WoofJS - making JavaScript learnable.* At https://stevekrouse.com/woof-d9adf2110fc6

Lewis C.M.(2010). *How programming environment shapes perception, learning and goals: Logo vs. Scratch.* Proceeding SIGCSE '10 Proceedings of the 41st ACM technical symposium on Computer science education, Pages 346-350.

Lewis C., Esper S., Bhattacharyya V., Fa-Kaji N., Dominguez N., and Schlesinger A. (2014). *Children's perceptions of what counts as a programming language.* J. Comput. Sci. Coll., 29(4): 123-133.

Matias J.N., Dasgupta S., Hill B.M. (2016). *Skill Progression in Scratch Revisited.* Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, 1486-1490.

Papert S. (1986). *Seymour Papert on Logo.* At http://el.media.mit.edu/logo-foundation/resources/onlogo/index.html.

Price T.W., Barnes T. (2015). *Comparing textual and block interfaces in a novice programming environment.* Proceeding ICER '15 Proceedings of the eleventh annual International Conference on International Computing Education Research , 91-99.

Scaffidi C., Chambers C. (2016). *Skill progression demonstrated by users in the Scratch animation environment.* Proceedings of the Conference on Computer Human Interaction (CHI), 1-39.

Weintrop D., Wilensky U. (2015 A). *To block or not to block, that is the question: students' perceptions of blocks-based programming.* Proceeding IDC '15 Proceedings of the 14th International Conference on Interaction Design and Children, 199-208.