Music Coding in Primary School

Luca A. Ludovico and Giuseppina Rita Mangione

Abstract This work stems from the recent reforms made to the Italian education system. On one side, a relevant aspect is the introduction of coding in primary school, where coding is seen as a way to improve learning processes in young students. On the other side, music education is considered a key element to promote the integration of the various aspects of personality: the perceptual-motor, the logic, and the affective-social component. After reviewing the state of the art, this work aims at defining a new discipline - called music coding - presenting the key elements of the two pedagogical approaches. A computer-based tool to foster the development of both coding and music skills in young students will be proposed.

Keywords Coding · Music · Education · Primary school · Pure Data

1 Coding for Young Students

Past research demonstrated that the adoption in primary school of techniques and technologies aiming at active, creative, and participatory training should be encouraged [1, 2]. Despite these evidences, for many years primary school education suffered from the lack of clear guidelines and innovative products able to generate evidence or disprove theories, as well as action scenarios related to this specific domain. Technology in school has been addressed as a form of training, and the educational goal - often enthusiastically emphasized - has become merely teaching how to use a computer.

L. A. Ludovico (✉)
LIM, Dipartimento di Informatica, Universitá degli Studi di Milano, Via Comelico 39, 20135 Milano, Italy
e-mail: ludovico@di.unimi.it

G. R. Mangione
Indire - Istituto Nazionale di Documentazione, Innovazione e Ricerca Educativa, Via G. Melisurgo, 4 - 80133 Firenze, Napoli, Italy
e-mail: g.mangione@indire.it

© Springer International Publishing Switzerland 2015

V.L. Uskov et al. (eds.), *Smart Education and Smart e-Learning*,
Smart Innovation, Systems and Technologies 41,
DOI 10.1007/978-3-319-19875-0_40

In 2014 the Italian Ministry of Education, University and Research (MIUR) announced the reform plan known as "La Buona Scuola".[1] One of the most relevant features was the introduction of coding as a curricular subject in primary schools. Several Western countries are in the midst of major changes to school curricula, since now the value of introducing topics such as programming, computer science and computational thinking is clear. For example, changes in the curricula have been introduced in 2014/2015 in the USA and UK in order to include basic programming experience in primary schools. These changes are stimulating public debate also in the scientific community, as discussed later.

In Italy, the children-oriented coding initiative - called the *Hour of Code* - is based on Code.org,[2] an international platform that lets young students move their first steps into the world of computer programming through easy tutorials and on line courses, often amusing and playful. Such a Web framework offers a number of courses, ranging from the basic concepts of Computer Science to the implementation and customization of a smartphone app. Lessons can be attended both on line and off line. At the moment of writing, Code.org's homepage states that more than 90 million students have tried the *Hour of Code* and more than 4 billions code lines have been written by young coders. In the framework of the Italian school system, this approach is extremely innovative. The guidelines of the mentioned reform suggest a change of perspective: some concepts and cognitive goals typical of Computer Science are seen as a fundamental asset for the cultural citizenship of the third millennium. The challenge and the bet is that approaching coding at an early age can improve logical and cognitive skills, with a positive impact on the future education and development of students.

The theoretical bases of coding go back to the pedagogical approach of *constructivism* formalized by Piaget, Vygotsky and Bruner [3]. As regards the introduction of coding and computational thinking in primary school, it is worth citing the ideas of Papert: while each discipline claims to push students to think, computer science achieves this result in an operative and concrete way. Papert's theories have been exposed in [4], adopting LOGO as the reference programming language for children's coding activities. An implementation of Papert's ideas requires a deep change in teaching, transforming the mere transmission of knowledge and skills into laboratory activities and structured projects aimed at encouraging collaboration and discussion. Needless to say, this implies not only the redefinition of school curricula but also ad hoc teacher training initiatives.

Opinions about the early teaching of coding spans from an enthusiastic support - problem-solving skills, innovative ways to explore concepts, etc. - to harsh criticism - development of industry-specific skills at the expense of a balanced growth of the individual. An example of the ongoing debate within the scientific community is provided by [5] and [6]. The former work stated the need for new creative and engaging learning experiences for children, with particular reference to job opportunities related to information and computing. However, as argued by the latter work, these

[1] Literally: "The Good School", https://labuonascuola.gov.it.

[2] http://code.org.

claims did not convince the entire scientific community. For instance, specific questions were raised about the importance for children to be prepared to work against the opportunity to enjoy a childhood without risks such as alienation from the real world and limitations in the development of imagination.

This debate had an impact on the review of primary school curricula where the so called *Hour of Code* had been introduced as a discipline [7]. A key factor for the updating of school curricula has been identified in [8]: "We teach kids how to use software to write, but not how to write software. This means they have access to the capabilities given to them by others, but not the power to determine the value-creating capabilities of these technologies for themselves".

The mentioned works provide a good overview of pedagogical motivations of coding. Its proponents usually allege one of the following reasons: (i) making students proficient in computer programming in general, and (ii) supporting the computational thinking that will be useful to student also in other educational contexts. When educators and computer researchers are requested to explain the advantages and opportunities that coding offers to young students, they go beyond the consideration that children can become good programmers in the future. Mitchel Resnick, one of the developers of Scratch at MIT in Boston, asserts that children must learn to create through digital technologies, and not only to interact with them [9]. When he describes the motivation that led to the development of Scratch, he claims that the main goal was "to nurture a new generation of creative, systematic thinkers comfortable using programming to express their ideas".

In coding activities, students are exposed to *computational thinking*, a locution which involves the use of computer science concepts such as abstraction, debugging, remixing and iteration to solve problems [10–12]. According to many experts, this form of thinking is fundamental for K-12 students because it requires "thinking at multiple abstractions". Moreover, computational thinking is in line with many aspects of 21st Century competencies such as creativity, critical thinking, and problem solving [13, 14].

It has been demonstrated that computational thinking does not mean only the acquisition of computational skills, even if certain abilities are practiced and trained through programming. For example, [12] argues that programming environments for children can support the development of three dimensions of computational thinking: computational concepts, computational practices, and computational perspectives.

In conclusion, writing code means more than using a computer: it implies finding original ways to achieve a goal, being able to decompose a problem into simpler sub-problems, becoming familiar with concepts such as sequencing, feedback and abstraction. As a consequence, the ability to code is an important skill since it may transform children from passive technology consumers into active technology creators [9].

The word *coding* is currently used by many organizations that promote learning programming skills, e.g. Code.org, Made with Code, Code Club, CoderDojo, Black Girls Code, Codecademy, Code Avengers, CodeHS, and MotherCoders. The increasing interest in coding at primary school is encouraging the creation of new learning environments known as *Initial Learning Environments* (ILEs) [6]. Needless

to say, coding environments for children have to present ad hoc features, including gamification and simplified syntax. For example, the adoption of visual rather than traditional programming languages facilitates computational thinking in K–12 contexts because unnecessary syntax is reduced (e.g., the use of semi-colon and curly brackets), commands are closer to spoken languages, and intuitive techniques such as drag-and-snap are employed to link command blocks [15]. In this context it is worth citing visual programming languages such as Scratch, Toontalk, Stagecast Creator, and Alice.

Popular coding environments have made coding accessible to a large number of students and teachers. As a virtuous effect of interdisciplinary contamination, curricular competences and skills related to basic programming are becoming an element in support of other subjects such as mathematics, science and technology. Besides, K-12 programming tools are becoming increasingly important in the context of digital literacy experiences for creating, sharing and remixing digital resources [16–18]. An in-depth review of available products goes beyond the purpose of this work. For further details, [6] provides a critical review of 47 environments used to introduce children to programming concepts, from simple game design to algorithmic challenges.

Most of the environments described in the following support computational thinking through pure self-discovery, without guidance on the cognitive aspects of computational practices and computational perspectives [19]. In this case, programming experience may be non-educative as students are not actively reflecting on their experience. In order to solve this problem in computational practices and computational perspectives there is a need to define new environments where information processing (e.g., metaphor, cognitive conflict and mind-mapping) scaffolding (e.g., explicit marking of causal reasoning) and reflection activities (e.g., peer reviewing and self video recording) could be designed [15].

2 Music and Children Education

Music education and training for children require ad hoc techniques and methods as well as a specific review of school curricula. Recent scientific works show that an integration of multi-modal experiences based on activities such as moving, creating, playing, reflecting [20] support the development of a “symbolically fluent child” [21, 22]. The learning environment should be able to represent activity-oriented musical experiences, where students - properly sustained by scaffold elements - are involved in a process of music construction/deconstruction. For example, according to [23], scaffolding children’s early musical experiences and investigations, their engagement in the world of sound, their trans-modal redesign of known literature and song repertoire helps children establish strong, confident, vibrant, and creative identities in learning, communication, and performance.

Currently a new music pedagogy based on an integrated approach is emerging. Recalling the fundamental concepts of pedagogical activism by Dewey [24], the goal is enhancing that educational cross-component able to influence key aspects of the growth such as expressiveness, autonomy and sociality. Music is able to influence the construction of the child's personality because it promotes the integration of perceptual, motor, affective, social and cognitive dimensions [25] by relating basic aspects of human life (e.g. physiological, emotional and mental spheres) with the basic elements of music (e.g. rhythm, melody and harmony).

The abilities of listening, exploration and analysis are fundamental for the development of general meta-cognitive skills of the child, such as attention, concentration, control. In this sense, music is both an opportunity and a crucial educational strategy. For example, through music young students can develop the aspects of analysis and synthesis, problematization, argumentation, evaluation and application of rules. In addition, as regards the ability to read and understand, children have the possibility to train their transcoding skills, moving from the musical domain to the verbal language in order to describe what they heard [26].

The construction of a vertical music curriculum can be the first step in a process that leads to the recognition of the musical dimension as a key element of the training process. The Italian guidelines for school curricula released in 2012 state that music is capable of supporting the achievement of skills essential for citizenship rights. In an official report published by the Italian Ministry of Education, University and Research in 2014,[3] music learning is described as a global experience that must interact with other disciplines and connect to other modes of expression. Learning music is seen as a synthesis of heterogeneous processes of exploration, comprehension and learning, and as a laboratory based on voice practice, conventional and unconventional musical instruments, graphics, gestural and motor activities. The goals to achieve are the integration of different musical languages and the construction of individual and collective identities. Nevertheless, in the Italian education system music teaching is often plagued by numerous factors: traditional lectures, boring subjects, limited instrumental practice, lack of creativity.

In the digital era, new technologies and computer-based approaches can influence music learning and teaching processes. A recent and comprehensive review of this subject can be found in [27], a work that discusses a range of innovative practices in order to highlight the changing nature of schooling and the transformation of music education. Many researchers, experts and music teachers feel a pressing need to provide new ways of thinking about the application of music and technology in schools. It is necessary to explore teaching strategies and approaches able to stimulate different forms of musical experience, meaningful engagement, creativity, teacher-learner interactions, and so on.

The idea of the present work is applying the most recent pedagogical theories to coding and music teaching in primary school through commonly available technologies and free, open-source tools, as detailed in the following sections.

[3]Transmission of guidelines relating to ministerial decree D.M. 8/2011.

3 Beyond Instrumental Practice: Music Coding

We have listed a number of scaffolds to provide children with logical and cognitive skills, going beyond the technical capabilities of a good programmer or a skilled music performer. In order to achieve this goal, teaching Computer Science in schools cannot merely imply a passive use of technologies, as well as teaching Music in schools cannot merely imply instrumental practice. We propose to combine the pedagogical advantages of coding and music education in primary school thanks to *music coding*, a new discipline that couples algorithmic thinking, technological tools, and computer interaction with musical experience, creativity, and social processes.

Although less famous than "traditional" programming languages, software tools to achieve music and sound programming are already available and commonly in use. For example, Csound is an *audio domain-specific language* (audio DSL), namely a computer language specialized to the application domain of music. Originally written at MIT by Barry Vercoe and developed over many years, it currently has nearly 1700 unit generators. One of its greatest strengths is that it is completely modular and extensible by the user. Besides, Csound is a free software, available under the GNU Lesser General Public License (LGPL). Csound and its possible applications have been explored in scientific works such as [28] and [29].

Under many points of view, Csound could be a good environment to experiment with music coding. First, this DSL has primitives to implement control flow, e.g. labels and *goto* statements, subroutines, conditional expressions and constructs, loops, etc. Both the compiler and most related software (editors, front ends, etc.) are free, and the environment is cross-platform. Finally, the sound result can be perceived by the programmer even in real time, thus providing a prompt action feedback. Unfortunately, producing sound through this environment is difficult for a child, as well as for a musician without a specific computer training. In fact, Csound takes two formatted text files as input, encoding the *orchestra* and the *score* respectively. The former text file describes the nature of the instruments, whereas the latter sets notes and other parameters along a timeline. Finally Csound compiler processes the instructions in these files and renders an audio file or real-time audio stream as output. Even if graphical interfaces can be implemented, such a programming language is far from visual-aid techniques typical of children-oriented coding environments.

Under the perspective of ease of use and intuitiveness, a formal graphical representation of music processes could help. A possible answer is provided by the extension of Petri nets to music. A Petri net is a mathematical modeling language invented in 1939 by Carl Adam Petri for the description of distributed systems [30]. Extensions of Petri net formalism to the music domain have been described in [31]. Music Petri nets are a formal and graphical tool to represent music processes. Concepts such as sequencing, conditional structures, looping, concurrence, non-determinism can be encoded. Besides, net topology can be changed and the corresponding sound generation can occur in real time, providing a prompt feedback [32]. Nevertheless, this formalism is tricky and can be cumbersome in the description of a music piece when a very fine granularity is required. Moreover, available software tools are not advanced enough to allow easy interaction with music Petri nets.

In summary, the first two approaches present some strengths: (i) They are oriented to music programming and formalization, so they may encourage the computational thinking typical of coding; (ii) They can be used to simulate and teach typical programming concepts; and (iii) Required frameworks are free and cross-platform, so they meet the need of low-cost technologies and availability even in the context of primary school. Unfortunately, these environments have also some relevant drawbacks that make the applicability to children education very difficult. Csound requires writing some text code appropriately formatted, where syntax is difficult to remember and even to be explained to young learners. All this is far from visual programming, drag-and-snap, and other concepts in use in K-12 educational contexts. On the contrary, Petri nets offer a graphical notation and their mechanism is easy to explain (the evolution of a Petri net due to fire rules and token-based marking may even resemble a game), but it is hard to grasp the underlying music meaning.

For the sake of comprehensiveness, another framework supporting visual aids and formalizing connections among functional blocks is the Reactable, namely an electronic musical instrument with a tabletop tangible user interface that has been developed at the Universitat Pompeu Fabra in Barcelona, Spain [33]. In its original implementation, the Reactable is a round translucent table, used in a darkened room, and appears as a backlit display. By placing blocks called *tangibles* on the table, and interfacing with the visual display via the tangibles or fingertips, a virtual modular synthesizer is operated, creating music or sound effects. This tool, used also by professional artists, presents a visual interface that recalls a playful approach. Besides, the Reactable provides a collaborative way of making music, as mentioned in [34]. In addition to licensing fees, this framework cannot be considered a real coding environment, since the function of blocks is predefined and the operating logic is oriented to music production rather than to the formalization of programming concepts.

A compromise solution, presenting all the pros mentioned above and minimizing the cons, is the adoption of Pure Data, a framework commonly in use in the community of musicians, visual artists, performers, researchers, and developers. Pure Data is a free and open-source visual programming language originally developed by Miller Puckette at IRCAM [35] and currently supported by a wide and active community of users and developers. Its applicability to the music-coding domain will be discussed in the next section.

4 Music Coding in Pure Data

Pure Data (Pd) lets users create software graphically, without writing lines of code. Possible applications range from sound processing and generation to 2D/3D graphics and video. As a consequence, Pd is suitable for learning basic multimedia processing and visual programming methods as well as for realizing complex systems for large-scale projects. Pd is a so-called *data flow programming language*, where software modules called *patches* are developed graphically. Algorithmic functions are represented by *objects* placed on a screen called *canvas*. Objects are connected together

with cords, and data flows from one object to another through these cords. Each object performs a specific task, from very low level mathematic operations to complex audio or video functions such as reverberation, FFT transform, or video decoding. Moreover, Pd can interface sensors, input devices, and MIDI-compatible music instruments. Thanks to these features, it is possible to design and implement rich, complex, and student-tailored educational environments. As regards the graphical layout of Pd patches, their standard aspect could be hard to understand for children. In this context it is worth citing the GrIPD (Graphical Interface for Pure Data) initiative [36], a cross-platform software aimed to design custom graphical interfaces for patches.

Now let us analyze the points common to a coding environment and to a music-oriented programming language. Coding environments for children should foster the development of three dimensions of computational thinking: computational concepts, practices, and perspectives [15]. *Computational concepts* are the concepts that programmers use, such as variables and loops. *Computational practices* are problem-solving practices that occur during the process of programming. For the sake of clarity, examples include being incremental and iterative, reusing and remixing, abstracting and modularizing. Finally, *Computational perspectives* regard students' understanding of themselves, their relationship to others, and the world around them.

Only the first item is usually covered by general-purpose learning tools and environments for children. On the contrary, a suitable music coding environment can easily fulfill also the other goals. As regards computational practices, in Pd music production as well as sound synthesis can occur in real time, exploring fields such as improvisation and modularization. Besides, the programming environment can be set to encourage reusing and remixing of music materials, like in so-called *step sequencers*. Finally, as regards computational perspectives, let us underline that music production becomes a social activity when students play together. A music coding activity extended to a class not only provides children with self-consciousness, but fosters relationships to other children and the surrounding environment. Moreover, a music coding environment can implement features recalling social-game mechanisms to allow peer-seeking (searching for the helping hand of a friend) and peer-reviewing (asking for other students' comments). Consequently, in our opinion the adoption of Pd - or a similar music programming framework - may provide not only music skills while fostering creativity, but even teach computational thinking better than other "traditional"coding environments.

5 Conclusion and Future Work

Our research aims to define and consolidate a theoretical and applicative framework in order to update school curricula with music coding. This concept implies a comprehensive experience that matches the pedagogical purposes of computational thinking with the development of personality typical of music education and practice. As a result of this theoretical work, a set of Pd patches will be designed and

implemented. After their release, a validation stage is needed to ensure that such an educational tool may satisfy the user's needs, here intended both from the teachers' and from the students' standpoint.

In particular, educational validation will be based on the detection of ad hoc indicators and metrics, in order to measure how music coding is able to support the aforementioned axes of computational thinking: computational concepts, computational practices, and computational perspectives. The dimension of the control and experimental group in primary schools is currently under definition, and future agreements will be established with Italian Territorial Centers for School Inclusion (CTI), Territorial Support Centers (CTS) and the Italian Ministry of University and Research (MIUR).

References

1. Kellett, M.: Developing critical thinking skills in 10–12 year-olds through their active engagement in research. Teach. Think. Skills **14**, 32–40 (2004)
2. Schweinhart, L.J.: The high/scope approach: evidence that participatory learning in early childhood contributes to human development (2006)
3. Piaget, J., Beth, E., Brousseau, G.: Learning: creation or re-creation? from constructivism to the theory of didactical situations. A Critique Creativity Complexity: deconstruct. Clichés **25**, 19–33 (2014)
4. Papert, S.: Mindstorms: children, computers, and powerful ideas. Basic Books, Inc. (1980)
5. O'Dell, J.: Why your 8-year-old should be coding. http://venturebeat.com/2013/04/12/why-your-8-year-old-should-be-coding/ (2013)
6. Duncan, C., Bell, T., Tanimoto, S.: Should your 8-year-old learn coding? In: Proceedings of the 9th Workshop in Primary and Secondary Computing Education, ACM, pp. 60–69 (2014)
7. Armoni, M.: Designing a K-12 computing curriculum: the questions. ACM Inroads **4**(2), 34–35 (2013)
8. Rushkoff, D.: Program or be programmed: ten commands for a digital age. Or Books (2010)
9. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al.: Scratch: programming for all. Commun. ACM **52**(11), 60–67 (2009)
10. Wing, J.M.: Computational thinking and thinking about computing. Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci. **366**(1881), 3717–3725 (2008)
11. Ioannidou, A., Bennett, V., Repenning, A., Koh, K.H., Basawapatna, A.: Computational thinking patterns. Online Submission (2011)
12. Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking. In: Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada, Citeseer (2012)
13. Ananiadou, K., Claro, M.: 21st century skills and competences for new millennium learners in oecd countries (2009)
14. Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M., Rumble, M.: Defining twenty-first century skills. In: Assessment and teaching of 21st century skills, pp. 17–66. Springer (2012)
15. Lye, S.Y., Koh, J.H.L.: Review on teaching and learning of computational thinking through programming: what is next for K-12? Comput. Hum. Behav. **41**, 51–61 (2014)
16. Mills, K.A.: A review of the digital turn in the new literacy studies. Rev. Educ. Res. **80**(2), 246–271 (2010)

17. Hague, C., Payton, S.: Digital literacy across the curriculum. Curriculum Leadership **9**(10) (2011)
18. Ng, W.: Can we teach digital natives digital literacy? Comput. Educ. **59**(3), 1065–1078 (2012)
19. Grover, S., Pea, R.: Computational thinking in K-12 a review of the state of the field. Educ. Res. **42**(1), 38–43 (2013)
20. Young, S.: Time-space structuring in spontaneous play on educational percussion instruments among three- and four-year-olds. Br. J. Music Educ. **20**(1), 45–59 (2003)
21. Jorgensen, E.R.: Philosophical issues in curriculum. In: The new handbook of research on music teaching and learning, pp. 48–62 (2002)
22. Barrett, M.S.: Sounding lives in and through music - a narrative inquiry of the everyday musical engagement of a young child. J. Early Child. Res. **7**(2), 115–134 (2009)
23. Tomlinson, M.M.: Literacy and music in early childhood. SAGE Open 3 (2013)
24. Dewey, J.: Art as experience. Penguin (2005)
25. Willems, E., Podcaminsky, E.: Las bases psicológicas de la educación musical. Eudeba (1969)
26. Branca, D.: L'importanza dell'educazione musicale: risvolti pedagogici del fare bene musica insieme. Studi sulla formazione **15**(1), 85–102 (2012)
27. Finney, J., Burnard, P.: Music education with digital technology. Bloomsbury Publishing (2010)
28. Vercoe, B.: Extended Csound. In: Proceedings of the International Computer Music Conference, International Computer Music Association, pp. 141–142 (1996)
29. Boulanger, R.C.: The Csound book: perspectives in software synthesis, sound design, signal processing, and programming. MIT press (2000)
30. Petri, C.A.: Introduction to general net theory. In: Net theory and applications, pp. 1–19. Springer (1980)
31. Haus, G., Rodriguez, A.: Music description and processing by Petri nets. In: Advances in Petri Nets 1988, pp. 175–199. Springer (1988)
32. Baratè, A., Haus, G., Ludovico, L.A.: Real-time music composition through P-timed Petri nets. In: Georgaki, A., Kouroupetroglou, G. (eds.) ICMC|SMC|2014 Proceedings, Athens 14–20 September 2014, pp. 408–415. Greece, Athens (2014)
33. Jorda, S., Kaltenbrunner, M., Geiger, G., Bencina, R.: The Reactable. In: Proceedings of ICMC 2005, pp. 579–582, Barcelona, Spain (2005)
34. Kaltenbranner, M., Jorda, S., Geiger, G., Alonso, M.: The Reactable: A collaborative musical instrument. In: Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE'06, IEEE, pp. 406–411 (2006)
35. Puckette, M., et al.: Pure Data: another integrated computer music environment. In: Proceedings of the Second Intercollege Computer Music Concerts, pp. 37–41 (1996)
36. Sarlo, J.A.: GrIPD: A graphical interface editing tool and run-time environment for Pure Data. In: Proceedings of ICMC 2003, International Computer Music Association, pp. 305–307 (2003)