The Proceedings of the 13th International Conference on Game Based Learning
ECGBL 2019

Hosted By
University of Southern Denmark
Odense, Denmark

3-4 October 2019

Edited by
Lars Elbæk, Gunver Majgaard,
Andrea Valente and Md. Saifuddin Khalid
Review Process
Papers submitted to this conference have been double-blind peer reviewed before final acceptance to the conference. Initially, abstracts were reviewed for relevance and accessibility and successful authors were invited to submit full papers. Many thanks to the reviewers who helped ensure the quality of all the submissions.

Ethics and Publication Malpractice Policy
ACPI adheres to a strict ethics and publication malpractice policy for all publications – details of which can be found here: http://www.academic-conferences.org/policies/ethics-policy-for-publishing-in-the-conference-proceedings-of-academic-conferences-and-publishing-international-limited/

Conference Proceedings
The Conference Proceedings is a book published with an ISBN and ISSN. The proceedings have been submitted to a number of accreditation, citation and indexing bodies including Thomson ISI Web of Science and Elsevier Scopus.

Author affiliation details in these proceedings have been reproduced as supplied by the authors themselves.

The Electronic version of the Conference Proceedings is available to download from DROPBOX https://tinyurl.com/ECGBL19 Select Download and then Direct Download to access the Pdf file. Free download is available for conference participants for a period of 2 weeks after the conference.

The Conference Proceedings for this year and previous years can be purchased from http://academic-bookshop.com

E-Book ISSN: 2049-100X
Book Version ISSN: 2049-0992

Published by Academic Conferences and Publishing International Limited
Reading
UK
44-118-972-4148
www.academic-conferences.org
Programming to Learn in Primary Schools: Including Scratch Activities in the Curriculum

Augusto Chiocciariello and Laura Freina
Institute for Educational Technologies of the National Research Council, Genova, Italy
augusto@itd.cnr.it
freina@itd.cnr.it
DOI: 10.34190/GLU.19.070

Abstract: The Italian Ministry of Education is promoting the introduction of coding and computational thinking in compulsory school. While it is still unclear how the Ministry will reform the current curriculum guidelines to introduce computing, Italian schools have anyway reached a record level of participation in events like the EU Code Week and the Hour of Code. "Programming to Learn in Primary School" is a project we are conducting following Papert’s claim that “children can learn to program and that learning to program can affect the way they learn everything else”. The project is in its second year and involves all primary school grades, from 1 to 5. The children in grade 1 and 2 work with programmable play kits with tangible interfaces. From grades 3 to 5 the online Scratch programming environment is used. To become proficient in a new language (the programming language here), children need time to learn how to use it expressively and become part of a social context where the language is practised. So the Scratch online community is a perfect match. Children love sharing their work and remixing, as well as the social features of Scratch for adding likes and commenting on each other’s projects. All the grade 3 to 5 children in the project attend a weekly computer lab class, playing with Scratch. In grades 4 and 5, they work on individual projects during the first half of the school year; in the second half, they work in small groups on a common theme that the teacher selects from those studied in class (the European Parliament, hydro-geological risk, etc.). The project aim is to develop and validate a vertical curriculum for the introduction of programming in primary schools as an expressive new language. In the lower grades the focus is on becoming fluent with the programming language, while integration with curricular disciplines is sought in the last two years.

Keywords: scratch programming, computational thinking, primary school, game making, coding

1. Introduction

In recent years, Computational Thinking (CT) has been widely recognised by educational stakeholders as a key competence for an informed citizen of today digital age. “Learning CT is about learning to think like a computer scientist – developing a specific set of problem-solving skills that can be applied in any domain to creating solutions that can be executed by a 'computer' (machine or human). Elements of CT include concepts such as logic, algorithms, abstraction, pattern recognition, evaluation and automation. It also includes practices such as problem decomposition, creating computational artefacts (usually through programming), testing and debugging, and iterative refinement. Collaboration and creativity are broader twenty-first century competencies that take on a special flavour in the context of CT”. (Grover & Pea, 2018 – p. 35). Coding/programming provides a laboratory for teaching and learning CT - it makes CT concepts concrete. It can become a tool for learning, e.g. as a medium for exploring other domains or for self-expression (through the creation of digital storytelling and/or videogames). However, there is general consensus that CT actually entails more than coding/programming.

In Europe, Computational Thinking is being introduced into formal education. The specific modalities in which this is taking place vary in different nations, nevertheless it is generally agreed that the earlier it starts the better results can be obtained. Accommodating new topics in the curriculum poses a number of difficulties per se, not least that it implies taking hours away from other activities. Several countries embed CT across subject areas, particularly at primary level, while at secondary level CT is mostly included as a computing subject in its own right (Bocconi et al., 2016). In Italy, the Piano Nazionale Scuola Digitale sets the government agenda to improve digital provision in education. The current version of the document (MIUR, 2015) explicitly mentions computational thinking; a specific action is dedicated to programming as a way of bringing computational and logical thinking to all primary school learners. The main rationale for introducing computational thinking is to foster 21st century skills and to move students from being passive users to active producers of technologies. Recently a unanimous vote of the parliament has committed the government to finalize the introduction of computational thinking and coding at school by 2022 (Mozione 1-00117, 2019). While it is still unclear how the Ministry will reform the current curriculum guidelines to introduce computing, Italian schools have reached a record level of participation in international coding events: first at EU Code Week and second at Hour of Code.
In the present paper, a project spanning over the five years of primary school is presented. The project aim is to develop and validate a vertical curriculum for the introduction of programming in primary schools as an expressive new language enabling children to move from consumers of information and computer technology to makers. The activities that were organized are outlined, the collected data is described and some preliminary qualitative results are described. The project is at its second year and a first data analysis is currently being performed.

2. The project

“My friends in the developmental psychology business were cynical about whether anything that could significantly be called programming could be managed by children who had not yet reached the so-called formal stage of development, which means about junior high school age. I saw the question as more subtle because I was more aware of how much it would depend on what is meant by programming.” (Papert, 1993 – pp 170-171)

Seymour Papert was a pioneer in creating programming environments for children as “objects to think with” and tools for learning. He’s also recognised for being the first to use the expression computational thinking in his writings (Grover & Pea, 2018). However, computational thinking is not a central theme of Papert research programme, Jeanette Wing ought to be credited for starting and promoting today movement for computational thinking. Papert shares with Piaget the idea that children have their own views on things—which differ from those of adults—and that these views are extremely coherent and robust. Children are active builders of their own cognitive tools. “To Papert, like to Piaget, better learning won’t come from finding better ways for the teacher to instruct, but from giving the learner better opportunities to construct. In his view, students best learn when engaged over long periods of time in the construction of personally meaningful products or products they truly care about. Open-ended design projects, under teacher’s guidance, usually offer greater opportunities for students to actively engage, collaborate, and contribute.” (Ackermann, 2010).

The paper presents a research project, Programming to Learn in Primary School, which spans over several years, aiming at studying a possible integration of programming activities into the Italian primary school curriculum. The project follows Papert’s (2000) claim that children can learn to program and that learning to program can affect the way they learn everything else. In involving children in programming, we looked for activities that they are passionate about in today digital world. Observation and previous experiences showed that children are very interested in using music, images, animations, etc. and in mixing them in creative ways. Programming allows them to add interaction, therefore to create interactive non-linear multimedia stories. Another important item that interests children are video games: they frequently play video games, they know them well. Programming allows children to build their own games. We then searched for the possible connections with the school curriculum. Some elements can be found directly in the programming activities:

- when creating a nonlinear story, or a game, children have to deal with logical concepts, both at an elementary level when designing how the story can evolve, and at a deeper level using more complex constructs (ifs, and, or, etc.);
- especially when making games, some other mathematical concepts like variables, states, coordinates, etc. have to be considered;
- while working at their creations, children learn how to solve the problems they meet, even though often with some external help. Since this learning is contextualized, and it is relevant to their problem, they often put effort into understanding the solution and easily remember it and use it again in different circumstances.

We look at programming as an expressive medium, a language children learn in order to be able to create their projects and express themselves. In this approach, the teacher has the role of facilitating students’ learning, he is a coach helping students to find their way. He is not any more the centre of all teaching activities.

Primary schools cover a span of several years, five in Italy, along which children have the possibility to acquire and consolidate their computational thinking skills. This can be obtained thanks to a sequence of age activities and programming environments. Furthermore, to guarantee a continuity from Kindergarten to primary school, some activities have been organized with some kindergarten classes (students aged from 5 to 6). Due to the wide age span involved, two main groups were identified: from kindergarten to grade 2 and from grade 3 to 5. In the following the two will be described separately.
3. From kindergarten to grade 2

The school we are working with is part of an “Istituto Comprensivo”. In the Italian school system, a comprehensive institute groups primary schools, lower secondary schools and pre-primary schools managed by a single headmaster. This gave us the opportunity to experiment with age appropriate programmable play kits both at the last year of kindergarten (five-year-old children) and the first two years of primary school. This school year, two kindergarten classes and one first grade primary class are involved in the activities. The inclusion of the second grade is planned for the next school year.

Following the Logo turtle tradition of offering both physical and “virtual” programming environments we choose to use Cubetto and Scratch Jr. A Logo’s turtle has a position and heading (looking somewhere). Children can identify with the turtle thus bringing their knowledge about their body and how they move to program it (Papert, 1980). Cubetto is a little wooden robot in the form of a cube that can be programmed to move on a square grid (Anzoategui, 2017). Cubetto design is inspired by TORTIS a version of the Logo turtle endowed with a tangible programming interface for pre-school children (Perlman, 1976). In a tangible programming environment, children move physical objects to construct a program instead of typing commands on a keyboard or arranging blocks on the screen. To program Cubetto children arrange a sequence of coloured blocks on a board, a sort of remote control, that send the instructions to the robot. ScratchJr is a graphical programming language based on Scratch and redesigned for the unique developmental and learning needs of children in kindergarten to second grade. It allows young children to create their own interactive stories and games (Flannery et al., 2013).

We have started working with this age range this school year focusing our activities on Cubetto; we did some experiment with Scratch Jr only in one class. One hour per week is allocated to these activities.

The children, both five-year and six-year old, use Cubetto to:
- move Cubetto on a map – either using the map that comes with the kit or drawing their own, the children decide on tasks for Cubetto and then program it;
- draw on paper - by attaching a felt pen on the side of Cubetto, in such a way that as it moves it leaves a trail.

For example, basing the task on the “Little Red Riding Hood” story, the children worked in group to program two Cubettons, one dressed as the wolf, and the other as Little Red Riding Hood, to find the best path to reach Grandma’s house after they met in the forest. The wolf needs to find the shortest route to reach Granma quickly, while Little Red Riding Hood wonders along a long path looking for flowers. The group of children discussed the solutions and then tried them out programming Cubetto (Figure 1).

---

**Figure 1:** A kindergarten class working with 2 Cubetto robots
4. From grade 3 to grade 5

In this age range, we have moved on to programming through the use of the Scratch online programming environment (Resnick et al., 2009). A programming language for this age range ought to be specifically designed for children. Seymour Papert advocated that it should have a “low floor” and “high ceiling”; i.e. easy enough for a child to get started with, while allowing for complex projects over time. Logo was developed following those principles. Scratch adds the “wide walls” dimension, “[...] different children have different passions, we need technologies that support many different types of projects, so that all children can work on projects that are personally meaningful to them.” (Resnick, 2017). Programming can be considered as an expressive medium, a language children learn in order to be able to make their creations. But, as for every language, learning requires time and best results are obtained when this happens in a context where the language is already used. The Scratch online programming environment offers its users the possibility to interact, exchange comments, share projects. Users can examine others’ projects; some components can be copied or the whole project can be remixed (making a personal copy that can then be modified). This support learning the programming language in a social context where it is practiced, and used as an expressive medium (Resnick, 2017). Scratch provides teachers with features to manage student participation including the ability to create student accounts, assign and reset password, organize student projects into studios, and monitor student comments. The involved class teachers created accounts each student in their class, and supervised them. Students were invited to share their creations, as well as to try to search for samples helping them to solve a given issue or search for inspiration.

In an initial phase of the project, students were left free to explore and experiment. The aim, at that stage, was to introduce students to the Scratch online programming environment and to the basic coding concepts. Sometimes, experts offered the whole class a new concept and then let them try it to their wish. Other times, single students were helped to solve their specific issue. Newly acquired knowledge naturally spread within the class as other students got curious or had to face a similar problem and peers helped them. For example, while some students were busy creating a simple game, the concept of “cloud variable” was presented. These are variables that keep their contents across different game plays, they can therefore be used, for example, to keep track of the best score ever. Students were very interested and some of them immediately added the new concept to their game (see the variable “punti” in Figure 4).

In grade 3, students started using Scratch and had to shift from the previously used tools to the more complex and complete one. During their free exploration, they were usually supported by our suggestions: sometimes a small project which could be of interest was shown to them and, starting from the new ideas, some students started their own creations. Another important scaffolding that was offered and worked well were the Scratch Cards. Even though Scratch comes with on-line tutorials, children did find more convenient to use printed instructions. A Scratch coding card features step-by-step instructions for beginners to start coding. The front of the card shows an activity children can do, while the back shows how to put together Scratch blocks to program it. Several copies of the Italian version of the Scratch Cards were available in the school lab and students were free to use them.

Furthermore, new concepts were offered on demand. For instance, some grade 3 students were looking for a way to insert a score in their game. The concept was explained to them and included in their game. As other students from the same class saw the game with the score, they were immediately interested and the use of variables spread in the class.

In grades 4 and 5, they work on individual projects during the first half of the school year; in the second half, they work in small groups on a common theme that the teacher selects from those studied in class. In grade 4, after an initial “heat-up” period in which students were free to explore Scratch, a game contest was organized. Students decided to work either individually or in pairs, and each participant had to design and create a game that, at the end of the year, will be presented to students of another class in the same school. Apparently, this activity is not strictly linked to any curricular activity, but this is not the case. Several mathematical concepts are emerging naturally from children’s work, furthermore their ability in analysing and understanding a problem, as well as their ability to stay focused on the task are being trained. The need to use loops and logic to make their games work, helps them understand these concepts, try them in their games and verify directly the obtained results.
In grade 5, integration with curricular activities gets tighter. After the initial warm-up with Scratch, a subject was chosen by the class teacher and students, divided into small groups, studied a specific aspect of it. Students were organized into groups to allow them to face more complex problems than they would on an individual basis. Furthermore, a cooperative group work allows every student to participate actively by offering their specific skills to the development of the common project, involving all the students of the class. The focus moves from programming to the subject that is being studied. Programming becomes the tool through which students can communicate with their younger peers. But, while doing so, they have to face programming issues that, when solved, increment their knowledge and allow them to better express their ideas. For example, during the first year of the project, a class that was about to go on a field trip to Strasbourg and visit the EU parliament, was asked to study the history of the EU, its parliament, the architecture of the parliament building, and the city of Strasbourg. In addition, the students created Scratch projects to explain what they had learned to a grade 4 class that was also participating to the field trip. This year, we are replicating this approach with two grade 5 classes. One is working on the presentation of the city of Genoa (Italy) from a historic, cultural and landscape point of view; while the other class is working on the geological risk and weather alerts. Both classes are creating their Scratch projects with aim of explaining something to a younger class.

Students were free to choose how to express their ideas with Scratch. In general, two different approaches emerged:

- Interactive multimedia stories, in which images, music and sounds, movements and texts are assembled in an interactive manner.
- Digital games. Games belong to the students’ everyday world, they know them well and often choose naturally to use this strategy for their projects.

According to Burke & Kafai (2012), different types of projects naturally entail different programming constructs. An interactive story needs to change settings, synchronize events, manage animations. On the other hand, games require the use of variables to manage scores and levels, often clones, loops, a more complex logic, etc. The choice to make a game, usually implies that the student feels confident enough to face the needed complexity. In the class working on the geological risk, all projects included an explanation and then some form of game: either a test in which the player gets points for each correct answer, or some other kind of games. In a case, players had to choose how to recycle correctly some objects, in another they had to avoid rubbish from piling up at the bottom of a river. Some groups imagined rather complex adventure games, in which the characters had to collect the needed tools and make the correct choice related to the situation surrounding them.
5. Examples of students learning within the project

In the projects that were created by the students, there are a number of cases in which some elements pointing to their learning emerged during the programming activity.

For example, a student programmed a bear to make it walk all the way to its cave using perspective, see Figure 3. Since the cave was further away from the observer, the bear’s dimension was made slightly smaller at each step of the animation, giving the impression that it was walking away.

![Figure 3: Two snapshots of a project showing the bear getting smaller as it walks away](image)

As projects get more complicated students need to develop their debugging skills. While younger students often just abandon a project that is not working as they expected, in grade 4 several students demonstrated to have the ability to analyse the software and understand where the problem was. Sometimes they needed help to solve the issue, but usually the possible solution was understood and they were then capable of using it again.

For example, a student working on a maze game decided to add dots that, when touched by the character while moving would add points to the player’s score. Since there were 80 dots, the student decided to place them at random positions on the screen. This might result in some dots overlapping with the labyrinth walls, and this made it impossible to touch them without touching the wall (which would have taken the player back to the beginning of the game - Figure 4). This issue was discussed and the solution implemented together by the teacher and the student in the first level of the game, and then added to the other levels by the student.

![Figure 4: Dots positioned on the labyrinth wall and cannot be taken](image)

Another student spent a long time working on a labyrinth game trying to understand how to control the character so that it would not cross the walls. She spontaneously explains how the problem was solved and what she understood in the notes associated to her project: “I studied how to be able not to go beyond certain colours, also looking at other projects to help me, and I understood everything. For example, if pressing right arrow Boh will change x by 3, then just make a script that in that particular case if you touch the red colour will change x by
-3. This must be done for all the arrows, but for up arrow and down arrow you will need to use y instead of x. And then the minus should be used depending on the direction of the arrow. These are small tricks on the Cartesian plane that, in my opinion, are essential and fun!” (Figure 5).

**Figure 5:** BOH GAME’s project page

6. Data collection, evaluation and assessment

The second year of the project is nearly over, and we are starting to analyse the collected data. Results will be available in the next months. In this paper, the data that is being collected is described and some initial, qualitative observations are reported.

All the projects that were shared by the participating students from their personal accounts are being analysed and categorized according both to some social indicators and some technical ones. In particular, for the social aspects, the use of instructions, notes and credits, as well as number of likes, favourites, remixes, and comments are counted. For technical aspects, the number of sounds, sprites and stages actually used, and the Scratch’s commands used. Given the large number of projects to analyse, we are using Dr. Scratch (Moreno-León and Robles, 2015) to automate the task of counting and classifying the Scratch’s command used.

Since each project is associated to its creation date, the analysis allows us to verify the use of certain programming concepts for each student along time. The change in time, as children progress towards a more sophisticated use of the programming environment, will be studied.

At the end of each school year every student completes a self-evaluation questionnaire, and semi-structured interviews are carried out with a selection of students.

The impact of the programming activities on the traditional school subjects will be evaluated at the end of a three years period, for those students who were actively involved in the project for at least three years. In the Italian primary schools, a national assessment test is taken in the month of May, in grades 2, 5. This will allow us to compare class average results at this test of the experimental classes with the corresponding results from classes from the same schools which did not participate to the project. Improvement on the test results would be an indication of the positive impact of the programming to learn approach advocated by the project.

7. Conclusions

The present paper describes a project spanning over several years for the introduction of Computational Thinking into the Italian Primary school through programming activities. Grover and Pea (2018), when defining the practices related to Computational Thinking, lists: problem decomposition; creating computational artefacts; testing and debugging; iterative refinement; collaboration and creativity. These are closely related to that part of the 21st Century Skills that refers to the transversal skills: creativity, collaboration, communication, critical thinking, problem solving, and productivity (Voogt et al., 2013).
The described project aims at promoting creativity and self-expression, in particular through the use of a visual block-based programming language that offers also a social environment. Coding in small groups, through a sequence of trial and errors aimed at the development of a project, which is often is a digital game, fosters in students their creativity and the collaboration among peers for a common goal.

The project is currently at its second year, and some qualitative data is already available. All the projects developed and shared by students are being classified and the data will be analysed. As a result, an assessment of the programming concepts that are used at each school grade will be made with the aim of defining better the kind of activities that are best carried out at each age range. Furthermore, a close integration with the other school subjects is sought through the organization of project-based activities, to be done in small groups, related to some contents studied in class. This allows the use of coding as an expressive tool to be managed directly by the class teacher during the normal school activities.

References


Perlman, R. (1976). Using computer technology to provide a creative learning environment for preschool children. MIT AI Lab Mem No. 360/Logo Mem No. 24, Massachusetts Institute of Technology. http://hdl.handle.net/1721.1/5784


Scratch Cards, [Online], https://scratch.mit.edu/help/cards