Programming for 3 rd graders, Scratch-based or unplugged?

Aaron Gaio

Programming for 3rd graders, Scratch-based or unplugged?

Aaron Gaio[1]

[1]University of Palermo, Department of Mathematics and Computer Science, Palermo, Italy;
aaron.gaio@dmi.unict.it

In this paper we describe a comparison between two different approaches to teach some algorithmic and computational thinking to children, mainly in 3rd grade. Children's learning is taken into main consideration and we want to analyze the difficulties students encounter using the different approaches. Before that, an introduction is done, describing the research framework and methodology, offering the background for this research and outlining the larger research project from which the paper is derived. We then describe the tasks used and look at some examples of the difficulties children face, on one side dealing with the problem of abstract thinking while programming, and on the other having troubles relating more practical activity with what the calculator does.

Keywords: Primary education, curriculum, computer science, programming, algorithms.

Introduction

Computer Science and algorithms in education are gaining more and more importance as the use of digital technologies is nowadays part of everyone's life. New educational trends are therefore emerging both from the computer science and mathematics education research community and from elementary and secondary school teachers (Franklin et al., 2015; Richtel, 2014). The question about how young children learn computer science is still a new area of research; and providing effective learning opportunities to K-5 students is a big challenge (Hills et al., 2015; Gelderblom & Kotze, 2009). Some good examples have been tried in the secondary school, while we feel that not much is present, at least in our country, in lower school grades. Topics in computer science and discrete mathematics are not clearly delimited in our curriculum and teachers are usually not aware that they actually could. We are thinking of our work as able to enhance the study of teaching and learning skills of mathematical practice through discrete mathematics problems, both general skills, such as reasoning and modeling, and skills particular to discrete mathematics, such as algorithmic and recursive thinking.

Background and context

Preliminary survey among teachers – relation with cryptography

We had a first survey, with results collected from about 150 teachers, mostly in service and quite evenly divided between primary, middle and secondary school. The survey was done with an online platform. The result analysis is mainly following a quantitative approach, the qualitative analysis was referred to the codification of some particular key words used by teachers.

Analysing the results, teachers, especially at lower levels, admit not to have the necessary knowledge to teach this in school. Question was about their previous experience in learning cryptography and graph theory, as well as the connection of these to mathematics and computer science. Some of these teachers see the connection between algorithms, cryptography and mathematics in general and computer science as quite necessary, while some don't have this idea clear in their mind. Also,

teachers were asked if they had any previous experience in teaching the topic or if "they would be interested in teaching some algorithm, cryptography and other discrete mathematics topic to students", and feeling from their answer is that this results can be taken as a first starting and promising point to make something of this into the national curriculum. A detailed analysis of these results is available in another article (Gaio & Di Paola, 2016, in press).

National Guidelines and teaching situation

The Italian Ministry for Education, University and Research published the current National Guidelines for the first cycle (kindergarten to middle school) of education (Ministero della Pubblica Istruzione, 2012). These guidelines are not any longer a detailed description of school curriculum to follow, but just want to provide concepts from which the single schools and institutes, and teachers, can take the basic goals and competences to reach. Some general standards are set with objectives for the educational achievements and learning goals. In the section talking about mathematics, there is a great importance given to reading and understanding texts with logical content, build lines of reasoning, having own ideas and defending and comparing them with others; a positive attitude towards mathematics, realizing how mathematical topics are useful in the real world. Algorithms and logical thinking as also referred to as important in the technology chapter of the guidelines, for all school grades. Following these guidelines, and our idea as well, "the first education cycle has a prominent role in the school curriculum considering the importance of this time in every student's life. Within this, the school attributes great relevance to the education and teaching methods that can fully activate energies and potentialities of every kid".

Research question

Our main general research problem lies therefore in a proposal to alleviate the substantial lack of activities in the national school curriculum about discrete mathematics and computer algorithms, especially for primary and middle school. Both in the school programs and in textbooks, activities of this kind are missing almost entirely, despite many agree that they can be really useful to improve the skills mentioned above.

The purpose of this specific paper is to deal with the introduction of programming reasoning to children as young as 8 or 9 years old. The question is whether *it is better to approach the subject with an unplugged approach and only later go on with computer-based coding or if it is ok to proceed using Scratch-based software and tools to serve the same purpose*. We do this by describing two different approaches which have been used in the teaching of these computer science and discrete mathematics topics. We will in particular analyze and focus on certain difficulties students encounter while using both.

Theory and methodology

This is an overview, referring to our whole project's methodology and background theory.

Background theory

Teaching methods follow the model of Realistic Mathematics Education (Gravemeijer, 1994) and Guided Reinvention of mathematics (Brousseau, 1997).

Guided Reinvention of mathematics is based on Hans Freudenthal concept of mathematics as human activity. Education should give students the "guided" opportunity to "re-invent" mathematics by

doing it. This means that in mathematics education, the focal point should not be on mathematics as a closed system but on the activity, on the process of mathematization (Freudenthal, 1973).

Realistic Mathematics Education (RME) is an instructional design theory which centers around the view of mathematics as a human activity (Freudenthal, 1991); "The idea is to allow learners to come to regard the knowledge that they acquire as their own private knowledge, knowledge for which they themselves are responsible."(Gravemeijer, 1994). The main goal is to develop a local (i.e. domain-specific) instructional theory (LIT) that will allow students to "[invent] the mathematics themselves" (Larsen, 2008). This need two steps: Step 1, in which "students are engaged in activities designed to invoke powerful informal understandings" (Weber & Larsen, 2008); Step 2, in which "students are engaged in activities designed to support reflection on these informal notions in order to promote the development of formal concepts" (Weber & Larsen, 2008).

Research methodology

The methodology we are going to use is that of design research or design experiments (Cobb et al., 2003; Barab & Squire, 2004; Brown, 1992). For the purpose of this thesis, the developmental approach is taken into consideration (Plomp & Nieveen, 2007); development studies function is to design and develop a, research based, intervention (Steffe, 1983) and constructing design principles in the process of developing it. The goal is to explore new learning and teaching environments, to verify their effectiveness; to develop somehow new methods, instruments and teaching actions to further improve in the field of problem solving and logical thinking, using somehow unusual topics as algorithms and cryptography are for primary school students. Doing this the goal is to contribute to the development of new teaching and learning theories, taking into consideration learning processes in the specific situation, with contents and goals clearly defined. Design research is quite appropriate in this situation, as we are facing a brand new experience in an environment that we need to analyze carefully, i.e. on a local scale, considering all the different elements in the learning environment. The intended design experiment will be a classroom experiment in which the researcher (or researchers) will cooperate with the teachers in assuming teaching responsibilities. On one hand, the teacher is a part of the design team and will be a key role in the development and reviewing of the activities, on the other, they have no previous knowledge and need a guide to experiment with this new experience and new content to present.

Design, tasks, analysis and results

As a first design step, based on theoretical framework and literature, two hypothetical learning trajectories were designed for the two different approaches. One approach is Scratch-based, and has been taken from the most popular book on curricular resources about Scratch programming in our country (Coding, DeAgostini publisher, Ferraresso, Colombini, Bonanome, 2014). The second approach was developed by our research team, taking idea and inspiration from the Computer Science Unplugged project (Bell, Witten, Fellows, 1998, 2015 review) and other related sources (Casey et al., 1992), with a development, after a preliminary teaching experiment, to better adapt the activities to the school level and local situation and norms. The two HLTs are taking into account the theoretical framework presented above, both in the choice of tasks (e.g. some tasks are chosen for their RME approach,others for the group and cooperative work students have to do, and so on) and in the way of presenting them to the classroom or students.

The tasks we are going to describe are just some of the many sequences of tasks that were proposed to various schools and age groups during the 2015/2016 school year in the bigger research project. In a design research paradigm (Plomp & Nievenn, 2007), the activities were tried out many times, always with an a priori analysis together with the teachers and with a retrospective look after each lesson.

Schools	Grades	n. of Classes	n. of Students	Approach
1,2,4	3	4	78	Unplugged
1,2,4	3,4	5	80	Scratch-based
3,5	3,4	3	54	Unplugged
3,5	3,4	2	39	Scratch-based

Table 1: Classes involved in different grades, with students numbers and curriculum used

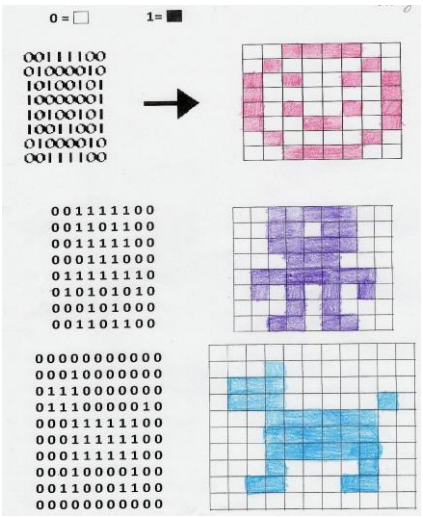
Scratch-based teaching and learning

As mentioned above, the sequence of tasks we called "Scratch-based" is taken from this Coding book, which is getting popular in our country's schools. Also we did use the M.I.T. official Scratch guide (Creative Computing, Brennan, Balch & Chung, 2014). It is following a similar approach to many school text books and even M.I.T.'s own guidelines on Scratch use and we feel it is good material for teachers. We did choose the tasks that are most popular among teachers already doing this kind of activity in their classroom, at least investigating the most popular in our area.

We did in particular choose tasks related to sequencing, selection and iteration. The goal is to have students learn basic ideas behind an algorithm (seen as a sequence of instructions), but also more complex concepts like selection instructions (i.e. do this only if something else happens) or iteration procedures. A sequence of tasks on those three topics were selected together with the classroom teacher and then tried out with the students during mathematics and technology lessons.

Unplugged approach, teaching and learning

Our "unplugged" sequence of tasks occupies 3 or 4 lesson slots of about one and a half to two hours and follows a brief introduction given on how computer works and binary numbers, in form of games (this was given also the groups using the other approach). Briefly describing the tasks, task 1 was an activity on paper, about binary image representation. Students had to color a grid which was provided with 0s and 1s and produce a drawing following the numbers. This task goal was about following instructions and beginning to understand how a computer transmits information.



Task 2 was about giving and receiving instructions. Students were divided in pairs and given a series of shapes and objects they could move on their table. One student (1) for each pair was to create a composition on his table; without looking at each other (physical barrier between the two), student 1 had to explain to the other how to reproduce the same composition with the objects and shapes. Children were required to be as precise as possible while the game went on, and to try to find out compositions that were harder to form. Only oral communication were left them, not to make them "correct" the other mistakes or looking at the other composition. Slightly different versions of the game were tried out, e.g. with just one student giving instructions to all others, or with different kinds of objects, even with just drawing something instead of moving objects, and so on.

The following tasks had the goal to make programming even more tangible for children. We wanted them to learn to give instruction as a calculator, through a path to walk on. One student (blinded) was the "robot" walking along this path on the ground and the others were the "programmers" having to give him instructions how to move to get to the end. We did this both speaking and then written. The written exercise does not give the possibility to correct the robot while you are actually giving the instructions, kind of how a real computer program works. Finally, with some of the classes we went on to construct some more complex sequence of instructions, posing different games to strengthen the concepts, but always with similar goals.

Methods of video selection and analysis

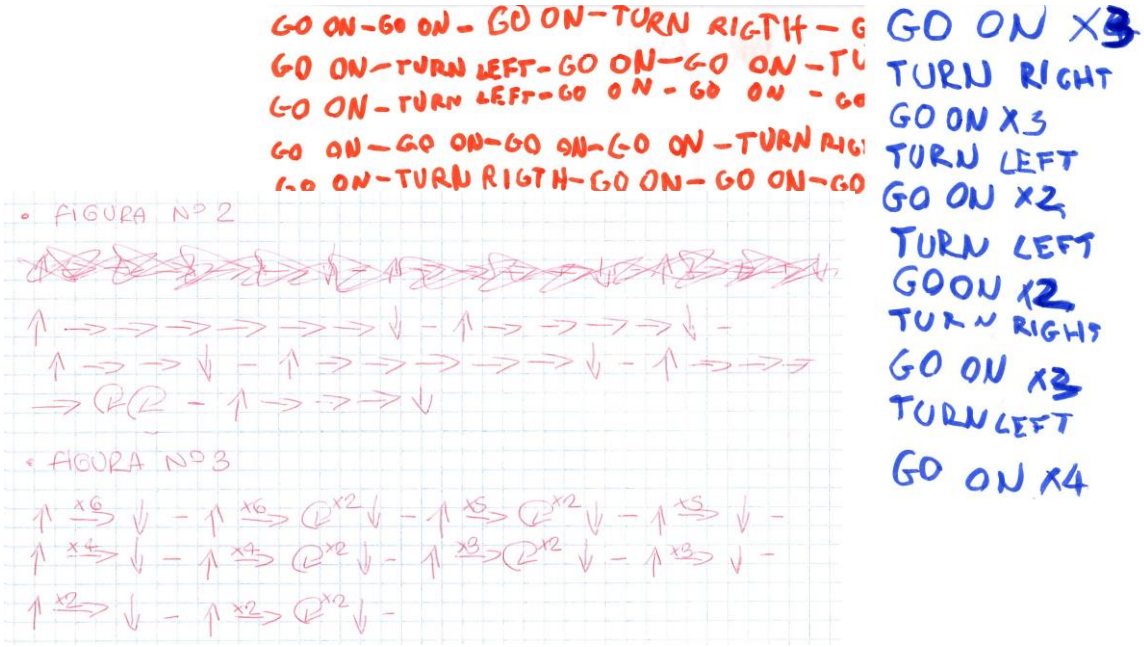
Analysis of the results is video-based, qualitative and fine-grained; both group activities and classroom discussion are recorded and we also have many of the transcripts, together with field notes, student's sheets, and interviews as other sources of evidence. As already said, focus is put on students' learning and thinking, in reaction to the different tasks proposed.

Following Zacks & Tversky theory, data is represented by events selected from the video recordings available. We used an inductive approach in video selecting, beginning with viewing the corpus in its entirety and focus on details later on. Indexing and summaries of videos, plus a content log, help in this process. Going on with the analysis some events which were particularly relevant were isolated. Although there are some recognizable recurring situation and choice of words we coded, our focus is more on a "play-by-play" description of these chosen events. We are, with this approach, analyzing selected episodes focusing on the same happening and constantly revising our finding and new hypothesis, as in Cobb and Whitenack (1996) with the involvement of "constantly reconciling provisional analytic categories with subsequent data and newly formulated categories".

Different difficulties emerging

The "events" we are focusing on does not pretend to show that one approach is better than the other, but which kind of, different, difficulties each of them can create in the children learning and thinking using the different approaches.

In the unplugged approach, children easily figure out what they are really and practically doing, drawing conclusions that they usually do not get in front of the computer. See for example the following figures where the transition from longer instruction in the first part to shorter instruction (switching to an iteration notation) in the second comes automatically. Almost every student quite naturally finds out that it takes a long time to write again and again the same instruction and is quickly asking himself if he might "make it somehow shorter". This is probably due to the fact that they were left free to develop their own language with arrows, and they feel they can adapt it to what is more appropriate and efficient for the situation. Videos showing these moments when they realize this fact has been isolated from the data.



Figures 5 and 6: Showing the transition from sequential instruction to iteration

Or, on the other hand, as an example, see the following short transcript from a video (in front of a Scratch set of instruction on the computer), where the students do not realize the usefulness of shortening a computer program to make it simple and more efficient:

Teacher: Why aren't you writing it in a shorter way (more compact?). You don't need to write an instruction 6 times, you could write "do this … times".

Student: Well, but what's the need for it? The computer is doing it anyways.

In these examples, students doing it unplugged quickly find out they are more efficient if they switch to an iterative mode of giving their instructions, while students doing it on the computer do not really realize this. On the contrary, they should learn one more command (or Scratch block) they do not already know to do it, so in the beginning it does not seem so appealing. From many events observed, quite surprisingly, this shortening is not immediate on the computer. Following our qualitative video analysis, in general, it seems that both selection and iteration instructions do not come naturally in

the Scratch environment as they do in an unplugged approach. Setting the activities and game in a real world scenario, especially at this young age, seems to give children a better idea of the advantages of iteration and the working principles of selection programming. Maybe creating some highly inefficient situation could force this process to happen in this case, too.

A second aspect to consider is errors done while writing a program. Analyzing error situation we can observe that it is actually easier for the students to spot the errors in a Computer-based environment. In the unplugged approach, sometimes, error fixing does not work at all, i.e. they cannot even spot the error if told that there is one. Our conclusion is that, as they are controlling their own game, they, more or less, unconsciously, get to a right solution even with a wrong set of instructions. On the computer-based environment, trying the program and making it running, given that the computer executes exactly what it has been told, students easily spots where the mistake is.

Another aspect we are facing at this young age is abstraction capability. As many references states (see Kramer, 2007), abstraction capability is the key to be a good programmer and to have future programming abilities. Abstraction is a very difficult process and Scratch helps a lot in this direction; on the other side, the unplugged approach makes activities somehow too distant from the abstraction of programming and makes it more difficult to children to relate what they are doing with what they will later do on the calculator, as some video excerpts from these moments show. Aspects of a real world mathematics surely can help the transition to the abstract world of programming (Futschek & Moschitz, 2011), but we have to be careful in the subtle connection between the two areas.

Conclusions

As conclusions, we could see pros and cons of both approaches, and we feel that obsessing over using just one is not the correct decision. Video and other data analysis show that there are aspects that ought to be dealt with in an unplugged way before writing them on the computer (algorithms, iteration processes and many others) and come really more natural to children if they use their own language, as can be seen in many "Aha! Moments" in the recordings. On the other side, it is really difficult for young children to relate the more real-world-oriented tasks to computer science, as we can see example when they get stuck in finding the connections. Future work will try to go into combining both approaches in a more comprehensive curriculum plan, creating new learning sequences that take both into account, which we will share with teachers and educators, in a relatively long developing process. Teachers willing to teach these topics are growing in number and they ought to be prepared for the challenge they will be facing.

References

Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground. *The Journal of the Learning Sciences, 13*(1), 1-14.

Bell, T. C., Witten, I. H., & Fellows, M. R. (1998). *Computer Science Unplugged: Off-line activities and games for all ages*. Computer Science Unplugged.

Brousseau, G. (1997, 2006 edition). *Theory of didactical situations in mathematics: Didactique des mathématiques, 1970–1990* (Vol. 19). Springer Science & Business Media.

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences, 2*(2), 141-178.

Casey, N., Fellows, M., Bell, T., Fellows, M. R., Witten, I., Fellows, M. R., ... & Fellows, M. R. (1992). This is MEGA-Mathematics. In *Proceedings of International Workshop on Parameterized and Exact Computation, IWPEC'09}*(Vol. 955, pp. 415-427). Los Alamos National Labs.

Cobb, P., Confrey, J., Lehrer, R., & Schauble, L. (2003). Design experiments in educational research. *Educational researcher*, *32*(1), 9-13.

Cobb, P., & Whitenack, J. W. (1996). A method for conducting longitudinal analyses of classroom videorecordings and transcripts. *Educational Studies in Mathematics, 30*(3), 213-228.

Derry, S. J., et al. (2010). Conducting video research in the learning sciences: Guidance on selection, analysis, technology, and ethics. *The Journal of the Learning Sciences*, *19*(1), 3-53.

Freudenthal, H. (1973). *Mathematics as an Educational Task*. Springer Netherlands.

Futschek, G., & Moschitz, J. (2011). Learning algorithmic thinking with tangible objects eases transition to computer programming. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 155-164). Springer Berlin Heidelberg.

Gelderblom, H., & Kotzé, P. (2009). Ten design lessons from the literature on child development and children's use of technology. In *Proceedings of the 8th International Conference on Interaction Design and Children* (pp. 52-60). ACM.

Gravemeijer, K. P. E. (1994). Developing Realistic Mathematics Education: Ontwikkelen Van Realistisch Reken/wiskundeonderwijs. CD-[beta] Press.

Hill, C., Dwyer, H. A., Martinez, T., Harlow, D., & Franklin, D. (2015). Floors and Flexibility: Designing a programming environment for 4th-6th grade classrooms. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 546-551). ACM.

Kramer, J. (2007). Is abstraction the key to computing?. *Communications of the ACM*, *50*(4), 36-42.

Plomp, T., & Nieveen, N. (2007). An introduction to educational design research. In *Proceedings of the Seminar Conducted at the East China Normal University [Z]*. Shanghai: SLO-Netherlands Institute for Curriculum Development.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67.

Richtel, M. (2014, May 10). Reading, writing, arithmetic, and lately, coding. *The New York Times*.

Steffe, L. P. (1983). The teaching experiment methodology in a constructivist research program. In *Proceedings of the fourth international congress on mathematical education* (Vol. 1, pp. 469-471). Birkhäuser: Boston, Massachusetts.