Is coding the way to go?

Violetta Lonati, Dario Malchiodi, Mattia Monga, and Anna Morpurgo*

Università degli Studi di Milano, Milan, Italy
http://aladdin.di.unimi.it

Abstract. Recently, several actions aimed at introducing informatics concepts to young students have been proposed. Among these, the "Hour of Code" initiative addresses a wide audience in several countries worldwide, with the goal of giving everyone the opportunity to learn computer science. This paper compares Hour of Code with an alternative, yet similar, approach which we believe is more effective in exposing pupils to the scientific value of the informatics discipline.

1 Introduction

Programming is at the core of informatics. This is occasionally forgotten in academic circles, but the importance of programming in the intellectual enterprise of computer science becomes evident if one looks at the tag cloud generated by the motivations for the ACM Turing Award winners[1] (see Fig. 1). Thus, introducing pupils to informatics through programming is a good opportunity to let them bite a real taste of the discipline [10,1]. Certainly this approach represents a better alternative to just exposing the students of primary and secondary schools to the use of computer applications, an unfortunate choice which impacted negatively on computer science education [16]. Driven by this belief, when in 2011 we decided to organize computer science enrichment programs for secondary schools using the algomotricity methodology [9], we designed some activities precisely focusing on programming, and in particular concerning the problem of guiding an automaton (a "robot": a long standing approach in the teaching of programming [13]) through a simple maze. Since 2013 these activities have consolidated into a short workshop [5] that met the interest of several schools in our town (about 50 classes have participated in two years). We directly conducted the first workshops, and specifically trained some young instructors to conduct them once they were well-established. To assess the outcome of the

Fig. 1. Tag cloud for ACM Turing award motivations

* The authors would like to thank the 'Istituto Comprensivo Ilaria Alpi', prof. Martina Palazzolo, and all the tutors that helped in conducting the workshops.

[1] Data taken from http://amturing.acm.org/alphabetical.cfm.

workshops we collected several materials, including interviews with some teachers, questionnaires filled out by pupils, recordings from focus groups. Henceforth this workshop will be referred to as *AlMa* (from "Algomotricity and Mazes"). AlMa is based on mazes, as is the starting example of the "Hour of Code" [12,17], an initiative launched in the US with a strong political endorsement (even President B. Obama wrote a line of Javascript to support it). The mission of the Hour of Code is "every student in every school should have the opportunity to learn computer science"[2]. Hour of Code's introductory proposal (*HoC* for short) is to capture the students with about an hour of coding games. Once students are motivated by HoC, the learning platform can be used (by teachers, but also by students alone) to start looking at more sophisticated computer science concepts and to foster a computational thinking approach. The HoC coding games are exceptionally appealing, featuring amusing characters and fascinating graphics. The platform is based on Blockly [7], a graphical programming environment inspired by Scratch [15]. The HoC website attracted a lot of interest (it claims more than 100 million completions of hours of code) and some positive reports have been recently published [11]. HoC quickly spread to several (180) countries; in Italy HoC is localized as "Programma il futuro" (*"Program the future"*, `http://programmailfuturo.it/`) and it is driven by the Italian Ministry of Education, Universities and Research. The apparent similarities between AlMa and HoC have forced us to reflect deeply on their differences, and since HoC certainly has a much wider impact than AlMa, we feel the urge to warn the community about what we perceive as a risk to direct pupils, again, towards the wrong target. We believe HoC is a very good intuition and a step that marks an important change of direction in the popularization of computer science; it is perfectly suited to attract pupils, to show them how fun informatics can be, to introduce them to coding. However, if the high-level goal is to show the actual essence and methodology of informatics, and to make students feel its *scientific nature* [8], then HoC can be misleading. Thus, in this paper we will describe AlMa in depth, comparing it with HoC, and we discuss what we believe is working better in our offer. The work is organized as follows: Sect. 2 describes AlMa and Sect. 3 illustrates the assessment of its outcome. Sect. 4 is devoted to comparing AlMa and HoC.

2 The Algomotricity Maze workshop (AlMa)

AlMa was offered within a wider set of workhops for pupils from 10 to 17 years old. About 50 classes have participated from the end of 2012, from several schools within the Milan District; AlMa was also occasionally proposed in other towns.

Goals. AlMa was developed having in mind the goal of showing the actual essence and methodology of computer science, with the final objective of capturing students to the challenges of a fascinating science, driving them to the *the scientific nature of informatics* [8]. The discipline is introduced in terms of activities focusing on the key themes of processing, automation and information,

[2] `http://code.org`

and promoting the use of some of the methods involved in computer science. From a more technical point of view, AlMa is meant as an introduction to the core of computer programming. Here, the syntactic issues are not the primary concern; instead, the activities are meant to help develop competences related to problem solving, computational thinking, exploratory analysis and scientific research.

Learning methodology. We designed AlMa according to a strategy we call *algomotricity* [9,3,4,5], since the activities focus on *algorithmic* concepts through *motoric* activities, and thus imply a mix of tangible and abstract object manipulations. Algomotricity starts "unplugged" [2] and ends with a computer-based phase to close the loop with pupils' previous acquaintance with applications [14]. This approach gives all participants the opportunity of exploring an informatics concept quite freely and lets them implicitly use the tools of scientific discovery, *i.e.,* formulating hypotheses to be validated by means of experiments. To foster discussion and peer-learning, all activities are performed by groups of pupils.

Description of the activities. AlMa first focuses on the task of verbally guiding a blindfolded person (a "human robot") through a simple path. Working in groups, pupils have to agree on the sequence of statements that a *driver* gives to a human robot. Initially they are allowed to freely interact with the robot, then they are requested to propose a very limited set of primitives to be written each on a sticky note, and to compose them into a program to be executed by the robot. Precisely, they are requested to use at most four different instructions: the constraint is enforced with sticky notes of four different colors at most, and by the requirement that, each time one of a certain color is used, it should always carry the same instruction. Also, they have the possibility of exploiting three basic control structures besides sequence (if, repeat-until, repeat-n-times). Groups may try their solutions as they wish and, when they are ready (normally after 30-45 minutes, depending on the pupils' age and motivation), each group is asked to execute its own program. After pupils have checked that their program allows the robot to correctly carry out the task, the conductor may decide to swap some programs, so that a program is executed by the robot of another group. This allows the instructor to emphasize the ambiguity of some instructions or the dependency of programs on special features of the robot (*e.g.,* step/foot size). In the last phase, which lasts between 20 and 40 minutes, students are given computers and a slightly modified version of Scratch. They are requested to write programs that guide Aladdin[3]'s lamp sprite through mazes of increasing complexity, see Fig. 2. The effect of some commands is illustrated by the conductor: the teacher refers to the first maze in order to show how to use the `move` n `steps` and the `turn clockwise` x `degrees` blocks; moreover she or he explains that the "walls" of the mazes are just black regions of the picture: the lamp can walk on them, but a correct solution is one in which this does not happen. Since the full Scratch platform could be confusing for someone who sees it for the first time (and needs to master it in half an hour), our version reduces the available blocks yet maintaining a rich spectrum of possibilities, so

[3] Aladdin is the name of our group: `http://aladdin.di.unimi.it`

that the students can focus on the motion and control ones. We also provide just one sensing block: `color` c_1 `is touching color` c_2, given with the hint that the lamp has the front in red and the exits have a distinct color, and we briefly explain how the sensor can be used to detect the exit. During the computer part, any working solution is accepted. To promote the use of control structures, a simple competition is proposed: the number of motion blocks used by each team (the lower the better) is recorded on the blackboard.

3 Assessment

To assess the outcome of AlMa, we collected the following materials and analyzed them in the spirit of grounded theory: (1) field notes written during the observation of some classes taking part in the workshop; (2) questionnaires filled out by pupils; (3) three focus groups with pupils; (4) interviews with some teachers. The assessment process involved 150 pupils and their teachers; all pupils attended the same suburban public school, who promoted the participation of all its 6th-grade classes to AlMa. Each of them filled out the questionnaires; the focus groups involved representative pupils from most of such classes.

Questionnaires. Pupils were asked to answer three open questions. (1) What did you like of the workshop? (2) What didn't you like of the workshop? (3) Is there something you feel you have discovered during the workshop?

We analyzed the answers and identified some recurring themes and strong concepts. Pupils claim to like: the fact that the workshop is both amusing and complicated/clever/challenging/engaging; the fact they have created/built something. They feel they have discovered: the importance of thinking/designing/figuring in one mind's what to do before doing it; the need for precision; that computers and other automatic devices do not work alone, but follow commands; that computer science is not only using computers; that informatics is a science; that informatics may be fun. It is worth noticing that such concepts emerged from all classes quite uniformly, thus they can be considered well-representative of the content and methodology of the AlMa proposal, and not depending on the different conductors or tutors who guided the workshops. We selected the most representative sentences from the questionnaires.
Thinking, designing, mind (answers to question 3) – "To think before doing otherwise you can make mistakes." – "You need to elaborate and set up your mind properly before acting." – "To plan the work in your mind."
Amusing and... complicated, clever, engaging (answers to question 1) – "Very amusing and complex." – "We played, but at the same time we reasoned."
Precision (answers to question 3) – "You have to be precise." – "Technological devices need very precise commands, in order to work properly."
Create, build (answers to question 1) – "When we had 'created' the maze." – "The part in which we had to 'build' a maze."

Focus groups. We proposed as discussion topics the main themes and concepts arising from the previous analysis. In order to activate the discussion, the selected sentences above were handed out and read aloud with the participants.

During the discussion most themes were recognized by all the participants. Everybody agreed on the importance of precision to avoid errors and/or risks for the robot, both during the execution of instructions, and when defining the instructions themselves (e.g., how many steps, which turning angle). We registered a unanimous agreement also on the need for reasoning before doing; in the discussions pupils repeatedly used verbs like *thinking, processing, preparing, foreseeing, understanding, solving, schematizing, agreeing*; or terms like *problem* and *logic*; or expressions like *organizing, ordering, putting together*, referred to both *ideas* and *instructions* (in the form of sticky notes or Scratch blocks). They confirmed that the tasks they had to carry out were fun and difficult at the same time, and stressed the fact that the challenge was part of the amusement, because "solving complex tasks is rewarding". However, when asked whether tackling with complex tasks is always amusing, they all clearly gave a negative answer ("I'm willing to use my brain, if the situation is enjoyable."), and pointed out that in this case the activities were fun per se. Words like *playing* or *game* were used to describe the activities, but someone felt such terms too reductive: "it was not child's play", "it was educational". Not everyone acknowledged that during the workshop a creative/building process took place. However, some pupils could establish links: to build is seen as a synonym for *to combine*, or *to put together*, hence this verb is associated with the process of combining sticky notes or blocks in Scratch; *inventing the actions* to write on the sticky notes was experienced as a creative process; for someone, even though the path was prearranged, but groups had to *create* the solution to go through it. Another topic proposed during the discussion is the perceived relationship between the workshop and the subjects taught in school. The concept of *precision* was immediately associated with technical drawing and mathematics; geometry was associated with the measure of length (number of steps of the robots) and turn angles. No spontaneous reference to science emerged. After the moderator suggested some hints, however, all pupils easily associated what happened during the workshop with the typical *observation-hypothesis-prediction-testing-analysis cycle* of the *scientific method*, and in particular with the concept of *experiment*. They told about several episodes when they made a hypothesis (for instance about how many steps were needed), designed a program/experiment, executed/tested it, and verified the correctness of their hypothesis. They also recalled that, when the experiment failed, they reviewed the hypothesis according to its outcome, and started the process anew: "the robot went too far, let's try with fewer steps!" And "when something goes wrong, you often discover something new that you didn't imagine before" (e.g., one is concerned about the number of steps, but finds out that also the turn angle is wrong). Such an approach was also used to choose among ideas proposed by different members of a group: some were tried and failed, while other survived to the experiment and were accepted in the final solution.

4 AlMa vs HoC

AlMa and HoC are apparently very similar. Indeed: (1) AlMa and HoC share the same high level goal: expose to informatics a variety of pupils (not necessarily involved in a computing curriculum), attract them through playful activities, and let them discover how fun and rewarding working with information sciences might be; (2) both are designed as a *first*, short, experience ("an hour" or so: AlMa lasts normally an hour and a half), possibly unrelated to a more structured study of the discipline; (3) AlMa and Hoc are conceived around the same theme: the problem of guiding an automaton through a simple maze. While similar (in fact Fig. 2(h) and Fig. 2(c) represent virtually the same maze), the context in which the tasks are proposed is rather different.

4.1 Algorithm, program, and code

To better illustrate the difference between AlMa and HoC, we discuss three terms that are sometimes informally used as synonyms. However their differences, while subtle, are crucial when one has to decide which one has the most potential to attract the creative energies of pupils to our discipline.

Algorithm is possibly the most noble term, with a long tradition (and several formal definitions that here we explicitly ignore): an algorithm is an *effective* procedure to reach, in finite time, a goal[4]. The key point is its effectiveness, a notion that could be clarified only by modern mathematics (Church and Turing above all): Euclid, Fibonacci, and al-Khwārizmī described their famous algorithms *on the assumption that their atomic steps were feasible and sensible.* A **program** is usually defined as an algorithm written in a programming language. In other words, in the post-Church/Turing/Von Neumann era a program is a procedure described in terms of the primitives provided by *a specific interpreter*. As the latter introduces specific syntax and semantics, converting an algorithm into a program can be a complex and creative task, a task largely independent from that of getting to an algorithm solving a specific problem. The recent parlance introduced a third term: **code**. What is then the difference with respect to a program? The word itself suggests a further reduction in the degrees of freedom, a constrained bijection between the procedure one has in mind and its machine implementation. In fact, this word seems well suited when one wants to emphasize the technological context of a program. Coding and programming are sometimes used as synonyms; we surely acknowledge that programming includes a coding activity, but we believe it entails, in general, a more complex endeavor.

4.2 The Hour of Code (HoC)

HoC was launched in 2013 as an activity planned in the Computer Science Education Week, in collaboration with big names of the software industry (Microsoft,

[4] One of the most rewarding activities we propose to teachers in our seminars on the didactic of informatics is the discussion of the notion of algorithm: we propose several procedures (cooking recipes, driving directions,...) and we ask why they are or are not actual algorithms.

Google, Apple, Bill Gates, Mark Zuckerberg,...)[12]. Its claimed goal is to "introduce computer programming to all students, to remove the veil of mystery that surrounds the field" [17], by also increasing the participation of women and other underrepresented students to computer science. Although the main HoC offer is based on an online activity, it does exist also in an "unplugged" version. Surprisingly, the unplugged alternative is rather different from the interactive one[5] is rather different: it proposes different tasks focusing more on methodological issues than on coding. To go beyond the first introductory hour, the `code.org` web site proposes also a 20-hour curriculum, with a mix of online and unplugged activities. In fact, it serves a lot of captivating videos and teaching resources, mostly about programming, based on Blockly and Javascript. In this paper, however, we focus only on the introductory part, intended to capture the interest of a vast audience of students who were never exposed to the fascination of the discipline. Moreover, the one hour format makes it comparable with AlMa which has similar goals of letting students meet computer science for the first time. The online HoC is entirely driven by the interactive puzzles: twenty mazes are proposed with increasing difficulties and by changing the constraints and the degrees of freedom of the "robot". The progression is the following (a sample of mazes is illustrated in Fig. 2(f)–(i)): **Mazes 1–5:** the students must code the solution by using the three blocks `move forward`, `turn left`, `turn right`; **Mazes 6–9:** a block `repeat` n `times` is added; **Maze 9:** the solution has to follow a constraint, given as a *grey* block which cannot be moved away; **Maze 10:** a block `repeat until at exit` is added (and the `repeat` n `times` is removed); **Mazes 11–13:** the solution has to use the `repeat until at exit` block, since it must contain no more than four blocks; **Maze 14:** a block `if path to the right` is added: it must be used correctly in a predefined scaffolding of four grey blocks; **Maze 15:** the block `if path to the right` has to be composed to meet the requirement of a solution with less than five blocks; **Mazes 16–17:** the solution should use as few blocks as possible (available: `move forward`, `turn left`, `turn right`, `repeat until at exit`, `if path to the right`); **Mazes 18–19:** the block `if path to the right` has now also an 'else' branch; **Maze 20:** in a predefined scaffolding of three grey blocks two selections are nested: the puzzle can be solved by choosing the appropriate statement to be put in the resulting three branches. Besides the constraints described above, it is also worth noting that each puzzle gives the solver just the subset of blocks useful to each quiz, although the subset is not necessarily minimal: for example, the `turn` block has a parameter `right` or `left`, but the player always has two blocks, one with the parameter set to right and one with the parameter set to left.

4.3 Differences

Problem solving or coding a predefined solution? Solving a problem is always a complex task: one has to distinguish the relevant pieces of information among irrelevant parts and build a model apt to reason about the solution. If a problem

[5] `http://studio.code.org/s/20-hour/stage/3/puzzle/1`

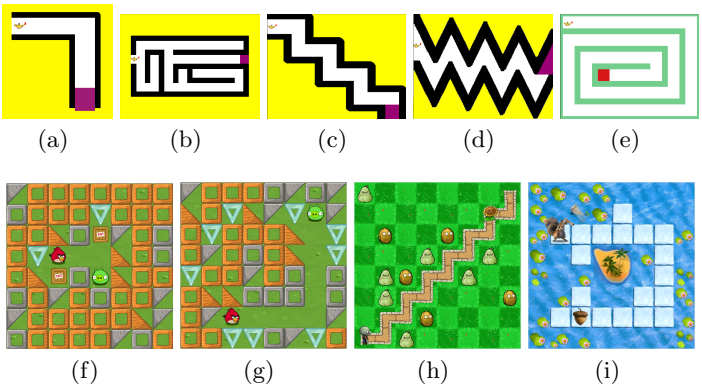

Fig. 2. Mazes used in AlMA ((a)–(e)) and in HoC, taken from `http://studio.code.org`: maze 3 (f), maze 8 (g), maze 12 (h), and maze 19 (i).

is given without noise, or already abstracted in a specific model, it is probably better considered as an *exercise* in a specific solving technique. In HoC, solutions are predefined (an easy trick to enable simple automatic checking) and almost suggested by the platform itself: the mazes are given within a grid, the robot moves with grid oriented steps, the number of blocks needed is given as a hint. Thus, the solver is left with the exercise of *coding* a solution by choosing the right blocks. When *the* solution is found, the system rewards the solver with a message giving the *number of lines of code* written so far (*"Congratulations! You just wrote 5 lines of code! All-time total: 6 lines of code. Even top universities teach block-based coding (e.g., Berkeley, Harvard). But under the hood, the blocks you have assembled can also be shown in JavaScript, the world's most widely used coding language"* [JavaScript version of the code follows].) Besides noting that most software engineers would agree that the number of "lines" is a misleading metrics, incidentally, it is worth noting that the generated Javascript code reveals some unfortunate design choices: for example, the `turn` block can be used to turn left or right by changing its variable part; instead, the generated code uses two different functions (`turnLeft()` and `turnRight()`), partially breaking the block metaphor and suggesting a questionable programming practice.

AlMa, instead, proposes little real problems. Students need to "formalize" them, they must find their own way to a solution, not just code a predefined one. Thus, the activity works on the interplay between algorithms and programs. When the pupils drive a blindfolded mate with a finite number of instructions, they reason on what is effective and feasible: and the power of the interpreter is in a large part a choice they explicitly do. Moreover, after the pupils themselves have checked that their program solves the task, when they compare it with others' programs, they discover that some of the assumptions they have made are not valid in the slightly different context of the other teams' settings. Comparing HoC to AlMa, a teacher said: "It lacks the first part which provides the link to

reality, to the difficulties of a real problem with its complexities and all its possibilities. Problem solving is the skill to reason computationally."

A riddle or a creative process? In most cases, the solution of HoC puzzles is unique, and the students must aim at guessing it. On the contrary, in AlMa any working solution is accepted. Pupils find out very soon how many different ways there are to accomplish the same goal, a first step in understanding that a program has also *non-functional* properties one might care of. At first, it may seem that students are left alone with Scratch in an intimidating free space of possibilities with just few clues to find their solutions, but the computer game comes after an even more open motoric part in which, however, they *invented* their instructions. What we found is that, after such a step, Scratch's blocks are a quite natural thing to use, and pupils can start trying to find the most similar ones to the commands they conceived in their sticky notes solution.

To be driven or to explore? AlMa aims at giving the students a meaningful problem to be explored in a suitably open context. We provide just a few restrictions designed to support their own inquiry. HoC proposes puzzles whose text mentions enough constraints to rule out all the solutions that do not use the intended blocks, especially when new blocks are introduced. In an interview, a teacher who had tried HoC and AlMa said: "HoC is very constrained, guided, it is more a nice tutorial to the visual framework and the use of blocks, than an actual opportunity for a problem solving activity". While the unplugged version of HoC advertises *computational thinking* as its main goal, the interactive HoC offers very few chances for exploring it. A teacher who proposed AlMa to her pupils last year (6th grade), this year invited them to participate to HoC and she reported they were facilitated a lot by they previous experience with AlMa: "the algomotricity experience introduced pupils to computational thinking, if they hadn't done it I should have figured out some introductory activity before presenting them the HoC proposal". The open-ended activities proposed in AlMa, indeed, encourage the participants to formulate original ideas. Moreover, the team setting forces the pupils to convince the other mates that their proposals work correctly: they need to describe them properly and devise a way to show the correctness of their hypotheses. Thus pupils happen to put into practice and experience the *scientific method*, even though they usually are unaware of this fact. Such an approach is not particularly useful when solving HoC mazes, where an easier (and faster) trial-and-error strategy is generally effective enough.

4.4 Discussion

All in all, if the final objective is to capture students to the challenges of a fascinating science, we believe HoC risks to give an incorrect first impression, only slightly different from the instrumental view so common in the teaching of informatics based on tools and computer applications. With the HoC approach, the scientific nature of informatics is not fully conveyed. Maybe it will be recognized later, if the pupils decide to go beyond the excitation of moving amusing characters. But if we want to show the actual essence and methodology of computer science, why not let pupils enjoy discovering informatics from the beginning?

References

1. Armoni, M., Meerbaum-Salant, O., Ben-Ari, M.: From Scratch to "real" programming. ACM Transactions on Computing Education (TOCE) 14(4), 25 (2015)
2. Bell, T., Rosamond, F., Casey, N.: Computer science unplugged and related projects in math and computer science popularization. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) The Multivariate Algorithmic Revolution and Beyond, pp. 398–456. Springer-Verlag, Berlin, Heidelberg (2012), http://dl.acm.org/citation.cfm?id=2344236.2344256
3. Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A., Torelli, M.: Exploring the processing of formatted texts by a kynesthetic approach. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education. pp. 143–144. WiPSCE '12, ACM, New York, NY, USA (Nov 2012)
4. Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A., Torelli, M.: What you see is what you have in mind: constructing mental models for formatted text processing. In: Proceedings of ISSEP2013. pp. 139–147. No. 6 in Commentarii informaticae didacticae, Universitätsverlag Potsdam (Feb 2013), http://opus.kobv.de/ubp/volltexte/2013/6368/pdf/cid06.pdf
5. Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A., Torelli, M., Zecca, L.: Extracurricular activities for improving the perception of informatics in secondary schools. In: Gülbahar, Y., Karata s, E. (eds.) Proceedings of ISSEP2014. Lecture Notes in Computer Science, vol. 8730, pp. 161–172. Springer (Sep 2014)
6. Glaser, B., Strauss, A.: The Discovery of Grounded Theory: Strategies for Qualitative Research. Observations (Chicago, Ill.), Aldine Transaction (1975)
7. Google: Blockly. https://developers.google.com/blockly (2011)
8. Hromkovič, J.: Contributing to general education by teaching informatics. In: ISSEP. pp. 25–37 (2006)
9. Lonati, V., Monga, M., Morpurgo, A., Torelli, M.: What's the fun in informatics? Working to capture children and teachers into the pleasure of computing. In: Kalaš, I., Mittermeir, R. (eds.) Proceedings of ISSEP2011. Lecture Notes in Computer Science, vol. 7013, pp. 213–224. Springer-Verlag (Oct 2011)
10. Meerbaum-Salant, O., Armoni, M., Ben-Ari, M.: Learning computer science concepts with Scratch. Computer Science Education 23(3), 239–264 (2013), http://dx.doi.org/10.1080/08993408.2013.832022
11. Nikou, S.A., Economides, A.A.: Measuring student motivation during "the hour of code^{TM}" activities. In: Proc. of the 2014 IEEE 14th Int. Conf. on Advanced Learning Technologies. pp. 744–745. ICALT '14, IEEE Computer Society, Washington, DC, USA (2014), http://dx.doi.org/10.1109/ICALT.2014.218
12. Partovi, H., Sahami, M.: The hour of code is coming! SIGCSE Bull. 45(4), 5–5 (Oct 2013), http://doi.acm.org/10.1145/2553042.2553045
13. Pattis, R.E.: Karel the Robot: A Gentle Introduction to the Art of Programming. John Wiley & Sons, Inc., New York, NY, USA, 1st edn. (1981)
14. Taub, R., Armoni, M., Ben-Ari, M.: CS unplugged and middle-school students' views, attitudes, and intentions regarding CS. TOCE 12(2), 8 (2012), http://doi.acm.org/10.1145/2160547.2160551
15. Team, M.: Scratch. https://scratch.mit.edu (2003)
16. The Royal Society: Shut down or restart? The way forward for computing in UK schools. http://royalsociety.org/education/policy/computing-in-schools/report/ (Jan 2012)
17. Wilson, C.: Hour of code—a record year for computer science. ACM Inroads 6(1), 22–22 (Feb 2015), http://doi.acm.org/10.1145/2723168