Extracurricular Activities for Improving the Perception of Informatics in Secondary Schools

Carlo Bellettini[1], Violetta Lonati[1], Dario Malchiodi[1], Mattia Monga[1], Anna Morpurgo[1], Mauro Torelli[1], and Luisa Zecca[2]

[1] Dip. di Informatica, Università degli Studi di Milano
[2] Dip. di Scienze Umane per la Formazione, Università degli Studi di Milano-Bicocca

Abstract. In order to introduce informatic concepts to students of Italian secondary schools, we devised a number of interactive workshops conceived for pupils aged 10–17. Each workshop is intended to give pupils the opportunity to explore a computer science topic: investigate it firsthand, make hypotheses that can then be tested in a guided context during the activity, and construct viable mental models. This paper reports about how we designed and conducted these workshops.

1 Introduction

Several voices have been recently raised to urge a new understanding in schools of the nature and scope of informatics. A report issued by the UK Royal Society on UK schools [14] has documented clearly how a focus on the instrumental value of Information and Communication Technologies (ICTs) failed to lead pupils and teachers to develop any real knowledge of computing sciences. While informatics and computational thinking have the potential to be very formative for pupils, and several core aspects of computing are sufficiently basic to be taught as a fundamental subject in all the secondary schools, among the general public—but sometimes also among the teachers who are responsible of conducting courses of informatics[1]—the scientific discipline is often blurred by the use of office automation tools, or the Internet communication facilities and their social impact. Although these activities might indeed need special skills, they can be presented and mastered without referring to computing at all. Therefore, to be able to show to pupils the appeal of a rigorous scientific discipline we believe it is important to focus again on the basics of informatics. Thus, we started devising some enrichment program activities aimed at presenting and discussing the core of informatics as the *"automatic processing of information"*.

We wanted to expose to informatics a variety of pupils of different ages, not necessarily involved in a computing curriculum or with a previous knowledge of

[1] We recently asked to a group of prospective teachers of informatics to define the discipline with a statement: 5 out of 17 identified informatics with communication technologies or other devices and a recurring theme was the emphasis on the improvement "of life quality". Common misconceptions among teachers are reported also by [13].

information or programming, therefore we developed an approach based on playful activities which imply a mix of tangible and abstract object manipulations: a strategy which we call *algomotricity* [6,4,5].

Table 1 summarizes the grades of the pupils to which we proposed the workshops. Each activity was designed to focus on one fundamental concept:

information What is information? How can symbols/numbers be used to represent it?

processing How can information be manipulated/changed in order to produce new knowledge?

automation Which manipulations can be performed by a *mechanical* interpreter? How this can be done?

Grade	4^{th}–7^{th}	8^{th}–11^{th}	12^{th}–13^{th}
information	Wikipasta	Human Pixels	Human Pixels (advanced)
automation	Mazes	Mazes (advanced)	
processing		Greedy Money (simplified)	Greedy Money

Table 1. Workshops proposed

The workshops are described further in Section 3, here we just recall their main focus:

Wikipasta In this workshop pupils are posed the problem of describing the typographic aspect of a text. By playing with pieces of pasta and other small objects, they are led through a game to the discovery of mark-up languages and then introduced to a lightweight "wiki" syntax. The final activity on the computer is about editing Wikipedia-like pages.

Human pixels After being shown a video of animations made in stadiums by coordinated soccer teams supporters (so called "human LCD"), pupils are asked to discuss how to set up a very simplified version of such animations. They eventually discover grids, sampling, resolution, compression and complete the activity by using a multi-view editor showing a picture along with different representations as a matrix of numbers.

Mazes In this workshop pupils are faced with the problem of guiding someone through a simple maze. Pupils first focus on the task of verbally guiding a *human robot* (a blindfolded mate) through a simple path. Initially they are allowed to freely interact with the robot, then they are requested to propose a very limited set of primitives and to compose them into a program to be executed by the robot, with the possibility of exploiting three basic control structures (if, repeat-until, repeat-n-times). After this, pupils are provided with a visual programming language (a simplified version of MIT Scratch) and are asked to write programs guiding a sprite through mazes of increasing complexity.

Greedy Money In this workshop pupils are requested to think about greedy strategies: they start by proposing an algorithm for the change-making problem with a set of coins that admits a greedy solution. Then they try to apply the same strategy to a scheduling problem and evaluate its suitability in finding the optimal solution.

Initially we tested such workshops in a lower-level and in an upper-level secondary school [4,5]. As a result, we could refine the design of these activities, and we prepared a set of two-hours workshops which we proposed to classes of 20–25 pupils. In 2013 a total of 26 classes attended our workshops, the activities are ongoing and we have currently planned 26 workshops also for this year.

The article is organized as follows: Section 2 discusses our methodological approach, Section 3 details the activities carried out in the workshops, and Section 4 draws some conclusions.

2 Methodological approach

All the proposed workshops share a common strategy, which we call *algomotricity*. As the name suggests (this neologism is a portmanteau combining *algorithm* and *motoric*), this approach exploits kinesthetic learning activities [1], having the aim of informally exposing pupils to a specific informatic topic, followed by an abstract learning phase devoted to let students build their mental models of the topic under investigation and a final computer-based phase to close the loop with their previous acquaintance with applications.

Our approach took inspiration from several papers in computer science education (for example [3,7,12,2]), and it is clearly rooted on the *Experiential Learning Theory* (ELT), specifically on *Problem based learning* (PBL). ELT defines learning as "the process whereby knowledge is created through the transformation of experience. Knowledge results from the combination of grasping and transforming experience. ... Immediate or concrete experiences are the basis for observations and reflections, these reflections are assimilated and distilled into abstract concepts from which new implications for action can be drawn. These implications can be actively tested and serve as guides in creating new experiences" [11]. PBL designs an educational environment based on experiential learning organized around the investigation, explanation, and resolution of meaningful problems. In PBL, students work in small collaborative groups and learn what they need to know in order to solve a problem [10].

In our workshops we designed *allosteric environments* [9] advocating a non-direct transmission of knowledge in favour of active work on part of the learner who constantly reworks her/his mental models in order to learn something new. Following this methodology, the teacher has the very delicate task of avoiding any predefined, generic way of presenting concepts. Rather, the instructor has to adapt to each pupil in order to give her/him suggestions interfering with her/his (mis)conceptions, setting up a so-called "didactic environment" [9]. This is typically done by putting pupils in new situations and assign them goals to reach whithout (or with just a few) hints about how to reach them.

Thus the teacher is a sort of mediator [9] who, having clearly in mind the goal of an activity, helps pupils toward this goal without forcing them to follow a specific path [2]. This requires a trade-off between free exploring and external constraints: the didactic environment should suitably limit the available degrees of freedom so that pupils can effectively and proficiently explore the solutions' space without either getting lost or having the feeling that there is only one *right* answer. Moreover pupils should have a real possibility to make mistakes.

This kind of set-up fits particularly well our learning goals, where pupils are facing a somehow new subject, as we are focused on *internal* aspects of algorithmics and information representation and/or processing, and not on learning how to use specific computer applications.

All activities ended with a computer-based phase, where pupils were confronted with specially conceived software in order to exploit what they had learned during previous phases. This is also necessary in order to match at least in part the expectations of pupils, who often identify informatics with the use of a computer.

Pupils were asked to work in groups, to encourage confrontation and peer-knowledge exchange, in all steps of the activities. The steps should be carefully designed according to Vygotsky's concept of *zone of proximal development* [15]. This requires a suitable choice of the difficulty of each step, both avoiding too gradual a path, resulting in bored pupils losing interest in the overall activity, and steep steps causing pupils to get lost. Moreover, the importance of the teacher's flexibility emerges also in this respect, because a same sequence could be adequate for some pupils and not for other ones. It is also very important to plan each step so that the activity makes sense *per se* in order to keep the pupils engaged [8].

3 Description of workshops

This section describes more in depth the workshops activities we proposed (except for "Wikipasta", which was extensively presented in [5]). Each of the workshops, lasting two hours and conceived for groups of approximately 20–25 students, was carried out in a room with enough space to ensure that pupils could move around in order to perform the tasks assigned to them.

The workshops start with a preliminary discussion devoted to let pupils express how they perceive informatics: at the beginning the conductor introduces the ethymology of the word "Informatics" and explains it as the *"automatic processing of information"* and pupils are asked to write on a sticky note their definition of the term on which the workshop is going to focus (see Table 1). All definitions are gathered together and clustered on a whiteboard, highliting common keywords. At the end of this process the conductor summarizes these keywords in order to let a group definition emerge from the notes written by pupils.

[2] Rather, the instructor should be able to exploit unexpected events to point out relevant issues not necessarily foreseen in the original design.

The core of each workshop consists of a phase where pupils engage in one or more game-like activities, followed by a teamwork session based on the interaction with expressly developed software tools.

A discussion phase ends each workshop. Its aims are to let pupils both reconsider their initial point of view about informatics and recognize a link between the game-like and the computer-based tasks.

3.1 Human Pixels

At the beginning of this workshop, pupils are shown a video of animations made by a group of soccer supporters in Korea[3]. In this video, each supporter wears a special shirt with different colors in its front and back part, where a third color can be shown by pulling two lateral flaps. When looked at from sufficiently afar, the group is perceived as an image and each supporter acts as a pixel. As supporters change the color of their shirts in sync and with sufficient velocity, the final result is an animation.

Pupils are then asked to form small groups (two or three persons) and to propose a method for showing a simplified version of such an animation, where each pupil of the class uses a black and a white paper sheet instead of the above mentioned shirt. The goal is to reproduce a simple message (*e.g.,* "Hello") through subsequent visualizations of the letters occurring in it. After each group has presented its idea, all pupils take a collective decision about how to organize the animation.

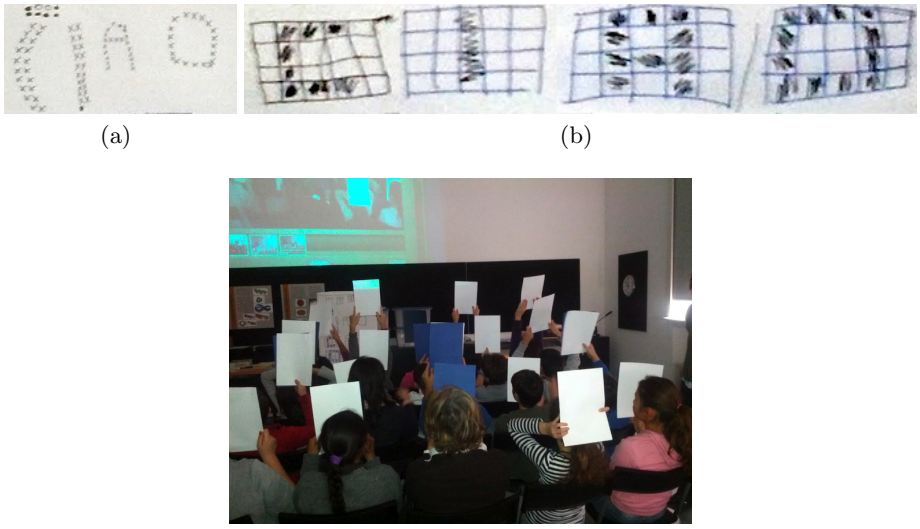
Although occasionally one of the proposed methods suggests to work with only black (or only white) sheets and changing each time the position of the students to let them shape the various characters (essentially discovering a representation in which every "pixel" is positioned relatively to another, as in Fig. 1(a)), pupils most of the times opt for:

1. building the equivalent of a black-and-white *bitmap* representation for each character occurring in the message (see Fig. 1(b));
2. positioning chairs in a rectangular grid;
3. assigning each student/pixel to a chair;
4. letting each student take note or memorize if the pixel s/he represents has to be set or reset in correspondence of the various characters, that is when s/he will have to use the black and when the white sheets.

Typically pupils also understand that some form of synchronization is necessary, and achieve it through a person who beats the time. A camera films the students so that they can test if the animation renders correctly (see Fig. 1(c)). Indeed, all pupils got a working animation, often also experimenting with variations such as animations with sliding characters.

As a final activity, pupils are introduced to a software especially conceived for editing bitmap images in a multiview interface. This interface shows a simple

[3] See `http://www.youtube.com/watch?v=tipHJlLUzNk`; we actually use a shortened version.



(a) (b)

(c)

Fig. 1. A sample of proposed "relative" (a) and bitmap (b) representation of the characters occurring in the animation. In (c) some pupils realizes a frame in the animation.

image along with different rasterized representations, such as a bitmap- and a RGB-based, respectively for black and white and for color images, and a compressed view based on a run-length encoding. Pupils are asked to play with the tool and discover how to modify an image by working on each of the different representations.

3.2 Mazes

This workshop focuses on programming. The kinestethic activity consists in letting pupils drive a mate through a given path. Pupils are divided into two or three groups of approximately the same size, and within each group the following actors are chosen:

- a *robot*, who, blindfolded, will have to follow the given path;
- a *driver*, speaking aloud instructions to the robot;
- a *care taker*, ensuring that the pupil playing the robot does not get hurt or fall.

The path is chosen according to the age of pupils, ranging from a simple L-shape to S- or 8-shape (see Fig. 2(a–c)). Moreover, the task includes the additional requirements that the *robot* grab one object placed in a fixed position and sit on a chair at the end of the path.

Pupils perform two rounds of such activity: in the first one they interact freely with the *robot*; in the second one they are asked to write a *program* to be read by the *driver* and executed by the *robot*, with the following constraints:

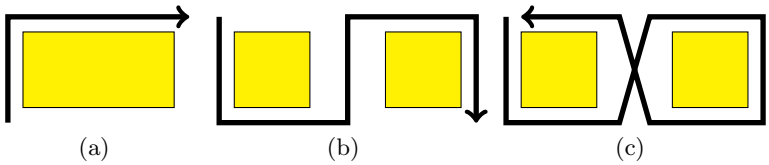


Fig. 2. Different paths for the algomotorial activity in the *Mazes* workshop: (a) L-shaped. (b) S-shaped. (c) 8-shaped. Paths are shown in increrasing order of difficulty.

- each group has to agree beforehand on the sequence of instructions that the *driver* will give the *robot*;
- they must work using at most four different instructions: students are given sticky notes of four different colors, and each time one of a certain color is used, it must always carry the same instruction;
- basic control structures (if, repeat-until, repeat-n-times) are available to simplify the program: the elementary semantics of these are explained;
- from a syntactic point of view, each instruction has to be written on a sticky note, stacking notes on a paper sheet in order to build a sequence of instructions, in case writing control structures around the sticky notes they refer to (see Fig. 3).

As a final step, after pupils have checked that their program allows the *robot* to correctly carry out the task, each of the programs set up by a group is executed by the *robot* of another group. Although groups have worked on the same maze, swapping programs has the effect of highlighting the following facts:

- programs are often specially tailored on some characteristics of the *robot* (*e.g.,* its step or shoe size),
- instructions are sometimes ambiguously or not precisely expressed (*e.g.,* turn left, without specifying the angle).

In the second part of the workshop, pupils are introduced to a simplified version of the Scratch programming language[4] with the task of moving an Aladdin lamp shaped sprite in virtual mazes of increasing complexity (shown in Fig. 4(a)–(e)). Pupils in pairs are challenged to write programs letting the sprite reach the exit of the mazes with the aim of minimizing the number of used instructions. Such minimality constraint lets pupils naturally apply specific control structures such as repeat-n-times and repeat-until, respectively in the mazes shown in Fig. 4(c),(d), and (e).

3.3 Greedy money

Groups of 2/3 pupils are asked to write down the algorithm they automatically execute when giving a change with the aim of using the smallest number of

[4] `http://scratch.mit.edu/`

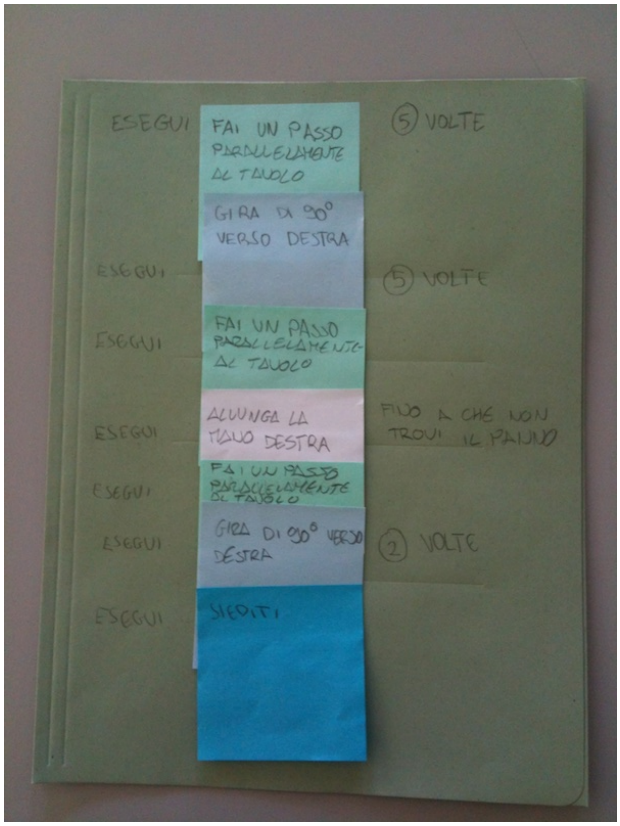


Fig. 3. A "program" containing sequences of instructions and control structures.

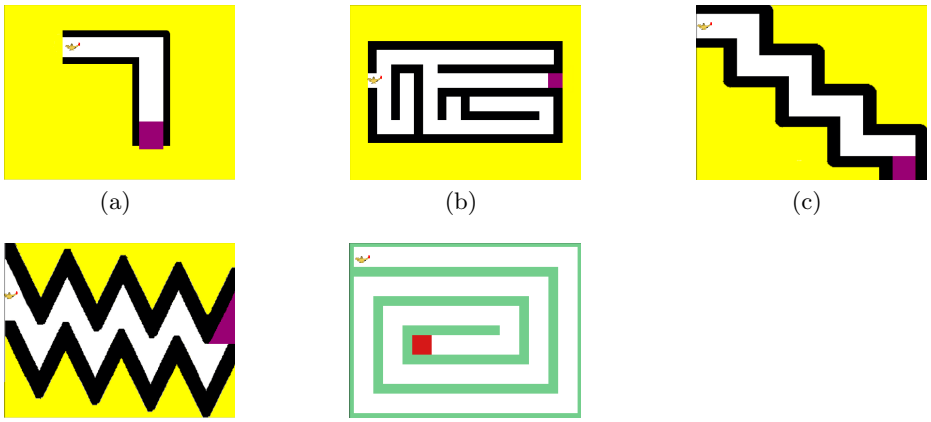


Fig. 4. Mazes used in Scratch. (a) L-shaped. (b) Standard maze. (c) Staircase. (d) Saw. (e) Spiral.

coins/bills. A set of play money can be used in order to test the algorithm through a step-by-step execution. Each group, in turn, describes its algorithm while the remaining ones test it. Most groups propose a procedure that manages to accumulate the change through subsequent additions of one coin or bill having the highest value yet not exceeding the residual change. Some use a more efficient approach exploiting the remainder of the division between the residual change and a coin/bill nominal value.

Thus the conductor can highlight commonalities, specifically:

1. the initial solution was the empty set;
2. coins and bills have been initally sorted in decreasing order of their nominal value;
3. coins and bills have been considered in such order, adding them to the solution until their value was higher than the residual change.

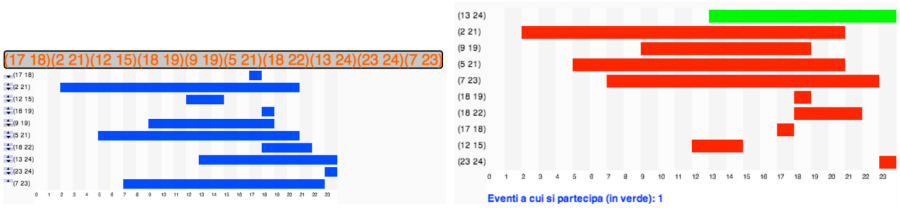
At this point, the conductor can generalize such approach to a more abstract procedure which builds the solution to a generic optimization problem by considering a set of objects in a given order, and for each of them deciding whether or not it has to be added to the solution according to a feasibility criterion. It should be emphasized that this constitutes an example of a *greedy procedure*, in that each object is considered only once for (possibly multiple) addition to the solution, without any possibility to remove objects previously added to the solution.

The second part of the activity consists in asking groups to apply such approach to a scheduling problem, namely that of maximizing the number of movies to be seen in a film festival whose program contains several, partially overlapping movies. Pupils are guided to find the analogies between this problem, the one concerning money change and the abstract description of a greedy procedure. Table 2 highlights such analogies: for instance, pupils tend to easily spot that movies, as well as coins/bills, play the role of objects. Analogously, the feasibility criteria are that of checking whether: i) a movie overlaps with other ones already in the solution, and ii) the considered coin/bill has a nominal value smaller than the residual change. Things change when considering the sorting order: the one based on nominal value naturally emerges in the first problem. On the second one there are several alternatives: starting or ending time , number of intersections with the remaining movies, or movies' length (either ascending or descending). Once all alternatives are clear, groups are asked to verify which criteria ensure the greedy solution to maximize the number of seen movies. More precisely, pupils are asked to find counter-examples to drop non-optimal criteria. A software supports the execution of this activity, generating at random a set of movies (cfr. Fig. 5(a)), rearranging them according to a chosen sorting criterion, and applying the greedy procedure (cfr. Fig. 5(b)).

Consider for instance the case shown in Fig. 5(b). With movies sorted according to decreasing overlapping numbers, finding a counter-example is easy. The greedy procedure would suggest to view only one movie (the highlighted one on the top), discarding all the remaining ones. Anyway, it would be possible to see four different movies: the fifth, the seventh, the eighth and the ninth from the

	Change problem	*Scheduling problem*
Objects	coins/bills	movies
Sorting order	nominal value, decreasing	several alternatives
Feasibility criterion	value not greater than the residual change	no overlapping between one movie and the ones already added to the solution

Table 2. Comparison between the scheduling and money change problem and the more general greedy approach.



(a) (b)

Fig. 5. The software showing a randomly generated set of movies (a) and applying them the greedy procedure according to a selected sorting criterion (b).

top. Students in different groups tend to find counter-examples for all criteria, except the one based on increasing ending time. The conductor can thus show an informal proof of the optimality of such criterion. In a final recap, students are warned about the fact that a greedy procedure does not always lead to the optimal solution. This happens for instance if using the non-optimal criteria for the film festival problem. It could also happen for the money change problem when considering a different set of coins/bills. Examples of such money system are the imperial British one or the one described in Harry Potter's books.

4 Conclusions

The power of informatics lives in its interplay between abstraction and concreteness: abstract ideas are implemented by concrete programs and real word machines. When presented in secondary schools, however, this power is often hindered by a predominant focus on using computer applications, thus informatics risks to be perceived as a bag of ready-to-use recipes, with almost no space for creativity, understanding and cleverness. *Algomotricity* is our attempt to present abstract symbolic manipulations in a very concrete way, but *without starting from computers.* Computers come at the end, just to close the conceptual loop with the previous acquaintance of the pupils with the ubiquitous ICT tools. We designed our workshops with a twofold goal:

– propose a methodological approach to informatics teachers, and

– present some core aspects of informatics to pupils of different grades and their teachers.

From a broader perspective, we aim at conveying a view of informatics as a scientific discipline, as opposed to the current perception of this field. We also paid attention in designing workshops requiring resources that are commonly available or easy to prepare (pasta, colored sheets, ...), and the software used is downloadable for free and runs on standard PCs. All classes participated with engagement in the proposed activities, collaborating, discussing different points of view, and showing satisfaction for their achievements. Many pupils admitted that their idea of informatics and of the job of ICT professionals was different before they attended the workshop. For example, the idea of informatics as a challenging discipline where creativity and collaborative work are often needed was new to many. A teacher in a lower secondary school, in an interview we made after a workshop, told us:

> "I did some experiments in teaching mathematics by concrete tasks, and I found them very important at this age, because pupils have great difficulties with abstraction, especially in the first year. The language of informatics is hard for my pupils because it is symbolic, abstract, it requires precision... They use computer applications by trial and errors, without reflection. From the workshop, I believe pupils learned that it is difficult to avoid ambiguities: for them it is difficult to put themselves in others' shoes..."

In fact, while informatics in Italy has been recognized as an independent academic discipline since the '70s, teachers with an informatic background are rather rare in non-vocational schools. Thus, while other abstract disciplines, like mathematics, have a certain tradition and established practices in secondary education, no such culture exists for informatics. Thus, we believe our workshops could represent a valuable experience to support teachers, and we want to use them as the starting point for a further action research.

Acknowledgments

We would like to thank the interuniversity research center MATEMATITA (`http://www.matematita.it/`) who hosted the workshops, and all the schools who partecipated to the activities.

References

1. Begel, A., Garcia, D.D., Wolfman, S.A.: Kinesthetic learning in the classroom. In: Proc. of the 35th SIGCSE TSCSE. pp. 183–184. ACM, New York, USA (2004), `http://doi.acm.org/10.1145/971300.971367`

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. Ben-Ari, M.: Constructivism in computer science education. Journal of Computers in Mathematics and Science Teaching 20(1), 45–73 (2001)
4. C. Bellettini and V. Lonati and D. Malchiodi and M. Monga and A. Morpurgo and M. Torelli: Exploring the processing of formatted texts by a kynesthetic approach. In: Proc. of the 7th workshop in primary and secondary computing education. pp. 143–144. WiPSCE '12, ACM, New York, NY, USA (2012)
5. C. Bellettini and V. Lonati and D. Malchiodi and M. Monga and A. Morpurgo and M. Torelli: What you see is what you have in mind: constructing mental models for formatted text processing. In: I. Diethelm and J. Arndt and M. Dünnebier and J. Syrbe (ed.) Informatics in Schools ISSEP 2013 — Selected Papers. Commentarii informaticae didacticae, vol. 6, pp. 139–147. Universitätsverlag Postdam, Universität Potsdam, Germany (2013)
6. V. Lonati and M. Monga and A. Morpurgo and M. Torelli: What's the fun in informatics? Working to capture children and teachers into the pleasure of computing. In: Kalaš, I. and Mittermeir, R.T. (ed.) Informatics in Schools. LNCS, vol. 7013, pp. 213–224. Springer, Berlin, Germany (2011)
7. Curzon, P., McOwan, P.W., Cutts, Q.I., Bell, T.: Enthusing & inspiring with reusable kinaesthetic activities. In: Proc. of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education. pp. 94–98. ITiCSE '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1562877.1562911
8. De Vecchi, G., Carmona-Magnaldi, N.: Faire construire des savoirs (2ème ed.). Hachette Education, France (2003)
9. Giordan, A.: From constructivism to allosteric learning model. http://www.ldes.unige.ch/ang/publi/articles/unesco_AG_96/unesco96.htm (1996), UNESCO Conference on Science Education 2000+
10. Hmelo-Silver, C.E.: Problem-based learning: What and how do students learn? Educational Psychology Review 16(3), 235–266 (2004)
11. Kolb, D.A., Boyatzis, R.E., Mainemelis, C.*et al.*.: Experiential learning theory: Previous research and new directions. Perspectives on thinking, learning, and cognitive styles 1, 227–247 (2001)
12. 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)
13. The CSTA Curriculum Improvement Task Force: The new educational imperative: Improving high school computer science education. Tech. rep., Computer Science Teachers Association (feb 2005), http://csta.acm.org/Communications/sub/DocsPresentationFiles/White_Paper07_06.pdf
14. 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)
15. Vygotsky, L.: Mind in Society: Development of Higher Psychological Processes. Harvard University Press, Cambridge, Massachusetts, USA (1978)