A low-cost tabletop game to collect learning analytics during computational thinking using unplugged or tangible activities

Sabrina Barnabé, Lola Denet, Mathieu Manrique, Divya Menon, Éric Pascual, Margarida Romero, Thierry Viéville

A low-cost tabletop game to collect learning analytics during computational thinking using unplugged or tangible activities

Sabrina Barnabé , Lola Denet , Mathieu Manrique , Divya Menon , Éric Pascual , Margarida Romero , Thierry Viéville

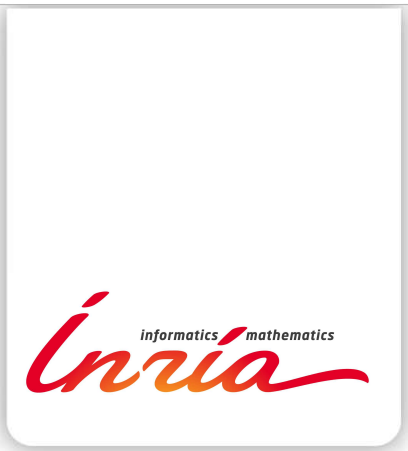
RESEARCH REPORT

N° 9379

December 2020

Project-Teams Mnemosyne and LINE laboratory

ISSN 0249-6399 ISRN INRIA/RR--9379--FR+ENG



A low-cost tabletop game to collect learning analytics during computational thinking using unplugged or tangible activities

Sabrina Barnabé [*], Lola Denet [†], Mathieu Manrique [‡], Divya Menon [§], Éric Pascual [¶], Margarida Romero [‖], Thierry Viéville [**]

Project-Teams Mnemosyne and LINE laboratory

Research Report n° 9379 — December 2020 — 39 pages

[*] SNJazur - Sabrina Barnabé <snjazur@gmail.com>
[†] Mnemosyne Inria Research Team - Lola Denet < denet.lola@gmail.com>
[‡] Laboratoire LINE, INSPÉ de Nice - Mathieu Manrique <mathieumanrique@gmail.com>
[§] Laboratoire LINE, INSPÉ de Nice - Divya Menon <divthefif@gmail.com>
[¶] PoBot maker space – Éric Pascual <eric.g.pascual@gmail.com>
[‖] Laboratoire LINE, INSPÉ de Nice - Margarida Romero <margarida.romero@univ-cotedazur.fr>
[**] Mnemosyne Inria Research Team et Laboratoire LINE - Thierry Viéville <thierry.vieville@inria.fr>

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Abstract:
We report on a new setup allowing us to collect learning analytics (LA) during computational thinking unplugged or tangible playful activities. We target the development of computational thinking (CT) competency, including the initiation to informatics (i.e., computer science and technology), with the goal to evaluate and analyze the development of CT. Collecting LA is mandatory in this case and if adaptive learning is targeted. While collecting LA during online interactions is rather straightforward, automatically collecting LA when manipulating tangible objects is more challenging, especially in a context where low-cost greenIT material is required.
The key idea here, contrary to usual "black-box" systems working (more or less) automatically, is to change the learning paradigm and involve the learner in the data collection, making the process transparent and allowing her or him to also learn how to learn. This is particularly pertinent here since we use Informatics tools in order to ... initiate to Informatics and CT. This means that we have to redesign the activity scenario including its didactic and revisit the underneath pedagogy, which turns to be an interesting and innovative challenge.

Key-words: computational thinking, learning analytics, unplugged activities, computational educational science.

Un montage de jeu de table à faible coût pour collecter des traces d'apprentissage pendant l'apprentissage de la pensée informatique avec des activités débranchées ou tangibles

Résumé :

Nous décrivons ici un montage original nous permettant de collecter des traces d'apprentissage (learning analytics (LA)) lors d'activités débranchées ou tangibles d'initiation ludique à la pensée informatique (computational thinking (CT)). Nous ciblons le développement de compétences en CT, y compris l'initiation à l'informatique (c'est-à-dire l'informatique en tant que science et technologie), dans le but d'évaluer et d'analyser le développement de la CT. La collecte de LA est indispensable pour évaluer cet apprentissage, avec comme champ applicatif l'apprentissage adaptatif. Bien que la collecte de LA lors d'interactions en ligne soit plutôt simple, la collecte automatique de LA lors de la manipulation d'objets tangibles est plus difficile, en particulier dans un contexte où du matériel à faible coût et tenant compte de contraintes écologiques est requis.

L'idée clé ici, contrairement aux systèmes habituels de «boîte noire» fonctionnant (plus ou moins) automatiquement, est de changer le paradigme d'apprentissage et d'impliquer l'apprenant dans la collecte de données, rendant le processus transparent et lui permettant également d'apprendre comment apprendre. Ceci est particulièrement pertinent ici puisque nous utilisons des outils informatiques pour ... nous initier à l'informatique et à la CT. Cela signifie que nous devons repenser le scénario de l'activité, y compris sa didactique, et revisiter la pédagogie sous-jacente, qui s'avère être un défi intéressant et innovant.

Mots-clés : pensée computationnelle, analyse de l'apprentissage, activités non connectées, science de l'éducation computationnelle.

This contribution is produced within the Inria AEx AIDE exploratory action and the ANR CreaMaker project.

Introduction

Learning Computational Thinking as the foundation of digital education.

Understanding how we learn is a key issue for improving education worldwide. This is especially true for transversal competencies sometimes referred as "21st-century skills" because some are rather new, such (i) computational thinking (Wing 2011). See also[8] (Lodi 2020) for a recent review. Transversal competencies related to (ii) cooperation are challenged when performed in ubiquitous and often asynchronous modalities with digital tools. Other transversal competencies such as (iii) creativity, (iv) problem-solving and (v) critical thinking are especially important in the present period for the development of citizens' transformative agency (Romero 2017). These five competencies are linked and must be considered in interdependence. For instance, computational thinking is intrinsically related to problem-solving and involves creativity in practice (e.g., creative programming). Computational thinking requires techno-creative activities to be developed and must integrate critical thinking development,especially when acculturating citizens of all ages to Artificial Intelligence.

As discussed in Romero and Duflot (2018), the development of computational thinking as the foundation of digital education seems to be effective considering unplugged activities (i.e., without a computer) for several well-understood within the cognitive embodiment studies and despite the experimental studies are not easy to establish (Romero et al. 2018). Nevertheless, a growing corpus of research is available regarding this topic (Menon, Romero, and Viéville 2019a) and more recently (Huang and Looi 2020).

The how and what challenges to evaluate learning activities

The barrier is the fact that it is not easy *how* to measure learning analytics during an activity with tangible connected or unplugged objects: The mainstream approach is to record a video and manually analyzed its contents, which is a huge work since observations have to be made on hundreds of sessions to obtain relevant results, as realized in Romero, David and Lille (2018). In such a context, machine learning assistive tools are only usable if there is enough data within a corpus of quality data. In addition, the analytical activity developed by a human may be subject to interpretation and the use of personal video data leads to heavy (but fully legitimate) ethics committee procedure due to the recording of personal data[9]. For limiting the ethical risks of personal identification of kids in the video recordings, the CreaCube protocol (Romero, Heiser and Viéville, in press) has limited the data collection to the hands of the learner, which is also collected through the tabletop dispositive within the fourth learning activity.

Beyond the measure challenge, is the question of *what* is to be measured. As studied in, e.g., (Romero, Lepage, and Lille 2017), it appears that the main challenge is to properly specify the task and the observables to be taken into account. This is indissociable of modeling the learner engaged

8The frenchy word "Informatics" instead of "computing" or "programming" is often used, because the required competencies extend beyond algorithmic and imperative programmation, but include information coding and data representation (from image coding to knowledge formalization), understanding of digital systems (e.g., machine architecture, networks, Internet) and related applications (e.g., the Web) all this being essential for technological hardware and software to make sense.
9Ideally, using automatic tools allows us to address this ethical issue, since the video stream is no more viewed by a human.

in the task, and our proposal, beyond the scope of this report, is to consider machine learning models, used in what is called numerical artificial intelligence, and ontology-based formalization, used in what is called symbolic artificial intelligence (e.g., for the semantic Web), to properly formalize, in the precise context of the task, her or his objective, knowledge tokens and behavioral rules, as explained in this position paper (Romero et al. 2020).
The key point, here, is that we not only want to propose some learning activity but also to evaluate it, to better understand how a given learner learns. Evaluating the process with the practical application of proposing adaptive and personalized learning "Parcours". Through this has been extensively developed for online activities (see, e.g., Banihashem et al. 2018) for a recent review and (Clement et al. 2015) for a well-studied successful example, in link with the formalization of curiosity and intrinsic motivation (Gottlieb et al. 2013), for real-life activities only a few studies have already addressed this issue as reviewed in (Menon, Romero and Viéville 2019a).

Considering not so common learning paradigms to take up the challenge.

Considering the need to evaluate the computational thinking process of the learner through a permanent monitoring has some profound consequences at the methodological level, as those raised by the Computer Support for Collaborative Learning (CSCL) community (van Leeuwen, Rummel, and van Gog 2019; Reimann 2009). Indeed, the learner is informed of the process, for obvious deontological reasons. This means that the relation to the activity is biased, and this could be a caveat or this could be of great advantages. More than informed about the fact the learning activity is monitored, the learner can be involved in this monitored learning, in the following sense. He or she plays with the machine. Not against the machine, but in cooperation with it: Helping the machine to monitor the activity, while the monitoring helps her or his to make explicit what to improve to succeed. This is a concrete way of learning to learn. The machine wins if it has properly monitored the activity and the learner wins if the activity succeeds. This kind of engagement of the learner seems, up to our best knowledge, never done before, and we are going to discuss how to implement this paradigm in such a performative learning session. It is however in link with the fact it is better to make explicit the learning objectives of a playful activity to engage the learner (see Menon and Romero, 2019) for a discussion).

At a more concrete level, proposing monitored unplugged activities (e.g., play with a pedagogical robot or even kitchenware) requires a non-trivial setup. Indeed such devices exist, but there are two key issues: Hardware and software, on one hand, and on the other hand human-power. Considering connected objects able to automatically report their position and configuration requires an investment beyond what is reasonable in a learning context (typically such system cost, with the functionalities proposed here, are at the order of magnitude of 20 to 50K€ plus an engineer full-time position) and one concrete objective is to break this barrier and allow a teacher to build their system, as soon as a maker-space (in the wide sense of a tinkering space) is available and can offer to the learner the appropriate affordances for his learning processes, in this case, the computational thinking process.

Bounding the research objectives of the present study.

For our research objectives to be reachable we have accepted the following restrictions. In the present configuration, the player is alone with the machine/artefact (tabletop configuration), but it is clear that learner-teacher interactions and inter-learners interactions are also crucial issues. Hopefully, lonely learning is also an important issue, as addressed here.

In the present configuration, only a short time (about one hour) learning period is considered, which again is restrictive, while of great interest on its own. A step further, learners' prerequisites, context and environment are not explicitly taken into account, activities being designed for “everyone”. This is yet another simplification, but as detailed in the sequel, we are only considering large audience introductive learning for which such simplification seems acceptable as verified in the literature

(Menon et al. 2019 ; Romero, Noirpoudre, and Viéville 2018).

Here we will neither consider facial expression recognition automatic recognition (see, e.g., (Nezami et al. 2019) for a recent contribution regarding student engagement with n good recent review of the literature), nor wearable sensor to monitor human activity (as reviewed in, e.g., (Mukhopadhyay 2015), including emotion-related processes using heart rate variability measurements or other biophysical measurements (see Prokofieva et al. 2019 for an example with application to education) for two reasons. On one hand, such issues are already addressed elsewhere while the challenges raised here are not. On the other hand, with such "easy" playful activity the young student engagement and stress seems to be a secondary issue as observed for instance, in Cassone et al. (2019), while this would not be the case if considering older students.

What is the paper about?

Given this general positioning, the paper is organized as follows. We first present in the following section the four chosen activities explaining the learning objectives, the activity to realize, and how the learner receives feedback in the function of the results to progress to the solution. The observables are made explicit, and the way the learner is engaged in learning analytics also. In the subsequent part, we describe the setup, focusing on the technological choices to produce low-cost hardware (we will explain why it is important), and robust software, both also being easily manageable, including rebuilding other activities, and will show that building such activity is by itself a very interesting activity regarding informatics. We then will report preliminary experiments with the proposed setup. We finally will explain the next steps of this long-term project.

Learning computational thinking first notions by playing

General learning objectives

The finality is to help discover and get initiated in computational thinking (CT), about problem-solving, creativity and critical thinking. We not only want to support but also to assess such learning. We primarily target kids from 7 to 10. Despite this primary target, the activity can engage other participants of laterages due to the transversal competences which are mobilized . We target beginners in CT, that simply know how to use a tap screen (e.g., a tablet) and know some tabletop games, to not to be disconcerted by the setup.

The didactic objective is to not only develop computational thinking competence in terms of algorithmic thinking (i.e., a sequence of instructions, conditional instructions, variable assignment and loops) but also understand some basis of information coding and data representation related to the analysis of the problem situation and the way to model a solution to devise a solution through formal systems (code) or physical systems (e.g. sensors, actuators, etc).

The pedagogical objective is to get convinced that "Yes, I can" understand such notions, easily affordable nowadays thanks to a real popularization[10] of learning methods and contents, that it can be playful to learn such things, beyond false ideas, and that it also helps understanding digital tools and technologies.
The fact this game helps both learning computational thinking and raise the interest of learning

10Let us very briefly quote at this stage https://scratch.mit.edu for creative computing, https://csunplugged.org for unplugged activities, and https://www.thymio.org for educational robotics, while elements are widely developed along the present text.

computational thinking is briefly made explicit before the game starts.

More precisely the activities have been developed within the #5c21 framework, as sketched out in Fig. 1 and considering the usual informatics initiation curriculum such as the British K–12 Computer Science Framework[11] (Curzon et al. 2014), as detailed in the sequel.

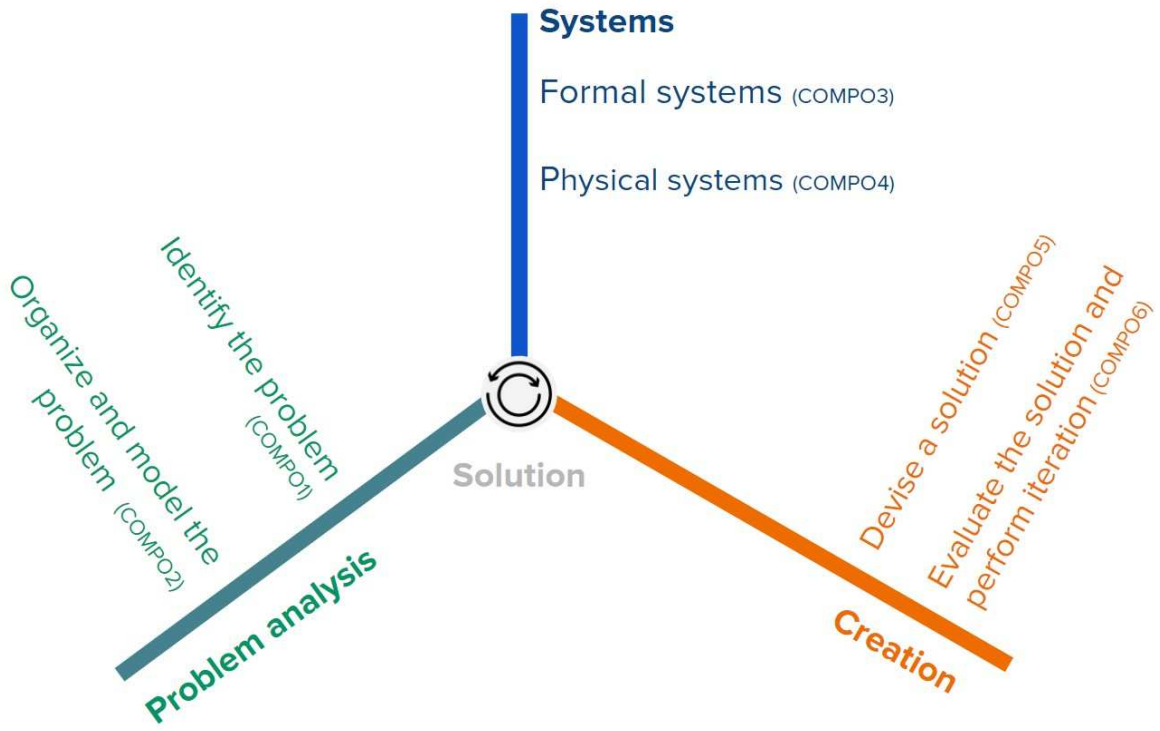


Fig.1 Six components of the CT competency within the #5c21 framework and related to the Collaborative Problem-Solving (CPS) of the Programme for International Student Assessment (PISA) 2015 (Romero, Lepage, and Lille 2017).

It is also a great opportunity to share some "scientific culture" which is understood at two levels:

- On one hand, build on the how-to (savoir-faire) skills acquisition to share some knowledge beans (savoir), more precisely use the mental construction elaborated during the how-to phase to build on and fix knowledge beans (here both in computer science and mathematics), as detailed in each activity description. In other words, we not only "play the game", but get an overview of the underlying notions. For instance, this allows the player to gain a concrete illustrative example of an abstract notion, or to experiment with some mathematical concepts with concrete (often surprisingly simple) object manipulation. At the pedagogical level, it is the fundamental way to introduce the notion of "computational thinking", making explicit abstract and general notions implicitly summoned and used during the playful phase. This will be stated for each activity.

- On the other hand, also considering a multi-disciplinary point of view, and varying the activity at the pedagogical level, it is time to discover some characters, pioneers of computer science, answering the double question: Who and How -- who found or invented what we discover and use nowadays? And how (in the historical meaning, i.e., in which context and circumstances, for which finality or goal, and with which means and tools (both material and intellectual) ? Even professional

11 Please refer to https://k12cs.org as a reference and https://tinyurl.com/y5fgadvm for a detailed description.

scientists often lack such scientific culture. Here we chose to invite men and women, from across the globe, and several historical epochs, showing that computer science is the emanation of centuries of multi-civilisations productions by the two half of humanity[12]. Knowing such "history" is always a good way to help personify and embody abstract notions in human history, to make it his or her history.

Scenario description

The activity is realized via a tabletop escape game, which method has been extensively studied in a previous work of the present authors (Menon, Romero, and Viéville 2019b), including game mechanics supporting a learning and playful experience in educational escape games[13] (Menon and Romero 2019). A player is invited to play a tabletop escape game with a progression of four activities, starting with a presentation and followed by an after-the-game period. Each activity corresponds to an activity room in a box with objects in a drawer that have to be put in a correct configuration on the activity room 2D plane within the box, as shown in Fig. 2. Activities include unplugged activities, tangible connected objects (cards, Cubelets[14]) and a mobile robot (Ozobot[15]).

The player plays alone with the tabletop, in interaction with the screen and the different objects.

The player always wins the game, depending on her/his performances he/she wins it without or with several clues. In particular, contrary to usual escape games, there is no time pressure, the game duration is "about" an hour, some actionable steps can be skipped depending on the previous result.

The activity is along a storyline scenario[16] based on computer science history. In a nutshell: « *I am Ozon the tiny robot girl and you know what? The fascinating grown-ups who created Informatics and I have been captured by evil monsters. They refuse to let you learn the story of this science and master all of these things, to reduce you to a digital slave. But I will share with you some secrets about Informatics and this will allow you to help us to escape from this "computational thinking" castle* ». Ozon is going to escape from a labyrinth in the castle garden, realize a magic recipe, send a secret encoded message and build a fantastic vehicle to escape with her new friends (i.e., scientists who have created Informatics). The realization is visible in Fig.2 and each activity has its small

12We indeed faced the problem that science history is -by the facts- biased in terms of gender balance and civilization balance, especially in computer science; we thus chose to be *illustrative*, but neither exhaustive nor representative.

13As developed in (Menon, Romero, and Viéville 2019b) and (Menon and Romero 2019), in escape games, after identifying a problem the learners need to organize and model the problem using the digital or tangible tools provided to them (identifying the problem, organizing and modeling the problem). They need to strategize the series of steps they should take to complete a task (sequencing). Players need to delegate tasks among themselves to identify the actions that can be done simultaneously (parallelism). The actions taken by them in the previous levels may affect the outcomes of the upcoming levels and the overall game outcome (conditional action). From the available resources, they need to decide on which tools or information will help them reach their objectives efficiently (physical systems). They can test their solutions and the visual representation of their outcomes can help them analyse their actions (debugging) and correct their approach (iteration).

14We use cubelets from https://www.modrobotics.com used for the problem-solving #Creacube activity from https://creamaker.wordpress.com.

15We are thankful to the https://ozobot.fr company who offered a few ozobots and allowed us to perform some reverse engineering to integrate this robot in our setup.

16Here is teaser text: «We are in 2020 and because of the use of digital the planet is in danger: Too much consumption of raw materials and energy, possible use of the Internet to question the humanist values of the peoples of the earth, instead to help develop them; to avoid this, let's go back in time, take up the history of IT, and redo the path that led to our digital society, so that in your life, and in everyone's life, digital technology allows us to live better and doing things well».

playful scenario, which will not be further detailed here, for concision[17].

When the game begins all four “rooms” are covered and locked, the player is introduced to the game via a small video and invited to play, after a consent form is signed (by the player or their guardian) and a pretest is realized. Then, the player enters the “room” (i.e., the first room cover is unlocked and by removing the cover the player discovers the labyrinth room).

Temporally the game is structured in activities and each activity is structured into steps: Presentation of the activity objective (both the gameplay and what is to be learned), iterative time of solution trial and feedback with more or fewer clues, achievement (including partial achievement), and proposal of the next activity. Each room has several possible activities, at different levels of difficulties, that are automatically skipped depending on the previous performances and the remaining time. At each trial-feedback, the player wins something: Either the activity itself, or some additional information (and encouragement), or some help.

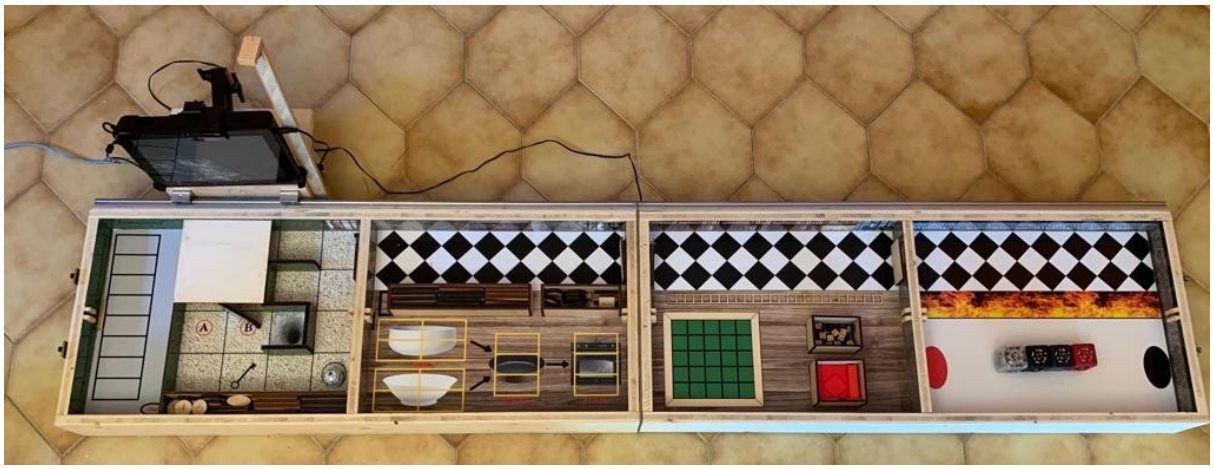


Fig.2: Top view of the Tabletop gameplay: From left to right, the (i) labyrinth, (ii) recipe, (iii) pixel art and (iv) cubelet activities. See text for details.

Evaluation methodology

The proposed setup aims to automatically measure the learning activity (through logs of the mediated activity) which consist of the object sensor trace and screen interaction, thus measuring the temporal sequence (also called longitudinal) log of the player activity within the different challenges proposed through the tabletop escape game.

Spatially, each activity is a rectangular surface on which the player can pick some objects and place them. The system camera takes a picture and parses the 2D scene, i.e., identifies pertinent objects at some given locations. The game state at a given time thus corresponds to a 2D configuration of the planar objects. This technological choice has several advantages: Automatic detection is rather simple to implement through standard image processing mechanisms[18], running on low-cost processors. More interesting, the mechanism is *easy to understand*, so that the detection does not appear as “magic” but something the player can afford and interact with (and even correct the log detector if a mistake occurs). A step further, the learning analytics is well defined as a logical structure (i.e., a so-called JSON structure) as a temporal sequence of an unordered set of 2D locations and labels, observed at a given time. The algorithmic implementation will be further discussed in the “Tabletop implementation” section.

In addition, screen interaction consists of looking at web pages with images, clickable “buttons”, and

17 It is available at https://aide-line.inria.fr where it can be browsed in detail.
18 Here we use the https://opencv.org open multi-platform and multi-language middleware.

small videos (which text is also readable). Player interaction is recorded as a displayed object click, on a given page, at a given time (as in standard web page learning analytics).

Temporally, as detailed before, the time is structured into activities, each activity having a presentation stage (that can be re-run at will) and a trial-feedback loop. Each feedback provides additional information, this being represented as a multi-scale finite-state automation. The rough scale has the presentation - trial - feedback states. At a finer scale, each trial-feedback state is labeled by activity information: The obtained result (e.g., the robot moves, or hits a wall, or disappears), and the next trial - feedback state to choose. A step further, at the task modeling level, additional information is to be added such as the precondition knowledge (e.g., understand the forward instruction) and the postcondition knowledge (e.g., the move size unit is one compartment).

The player learning benefit is to be evaluated from a small pre-test and post-test, and mainly after the game, after a pause (e.g., another day), by proposing small activities related to what has been learned. This qualitative benefit evaluation is performed via competence transfer evaluation as detailed in the sequel. More precisely at a semantic level, we consider

- *In-task observables*, measured as detailed in this section, while the content is going to be made explicit, activity by activity, recorded automatically.
- *Debriefing questions to the player*, on both the technical content and the related culture, recorded automatically.
- *Competence transfer activities*, the transfer activities are proposed after these session activities with related skills, performed manually with human interaction.

Activity 1: Programming is that easy.

For the first activity, a tiny mobile robot is programmed (Ozobot), as detailed in Fig.3a and Fig.3b, showing that the same setup leads to several activities with different levels of abstraction.

The robot itself can be programmed using the following instructions. Either left quarter-turn, right quarter-turn, or forward of 1, 2, 3 or 4 cells. This is indeed not the only functionalities of this Ozobot, but it has been restrained to a minimal setup for this basic activity.

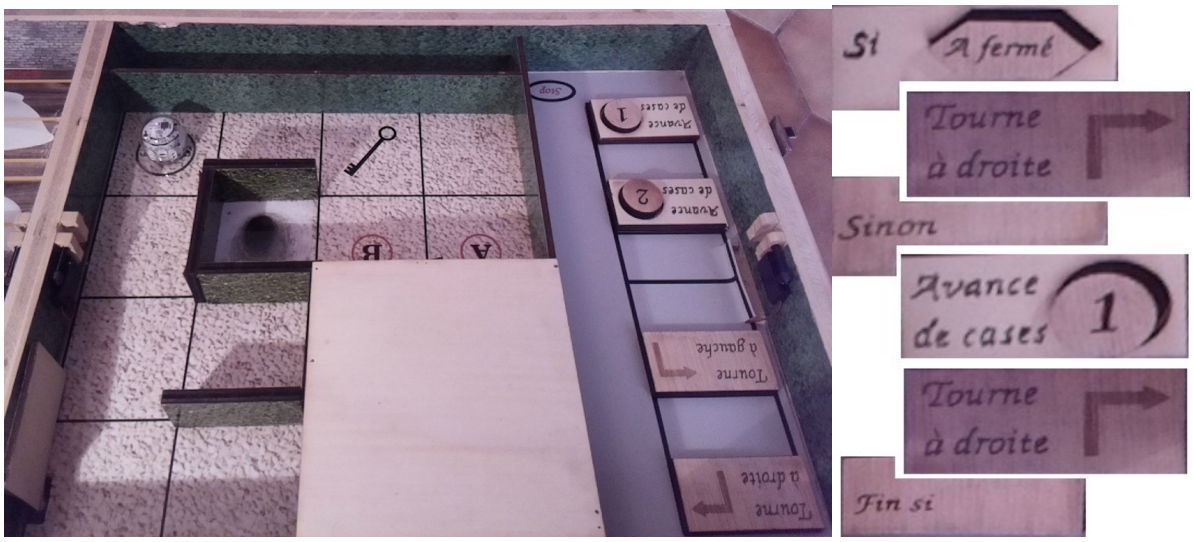


Fig. 3a: A tiny robot is programmed, instructions being materialized by handmade wooden cards, arranged in sequence and detected by the setup camera. The player places the card and has a “start” card to ask the robot to execute the program. The robot mission is to go on the “key” cell, to open the labyrinth gate, avoid falling

into the “devil well”, and cross over the tunnel, i.e., infer instructions including for the non-visible part of the route.
There are several “good” solutions (i.e., through the tunnel, or using the open-air path on the grid side, with the possibility to wonder what is the “optimal” solution.
If there is no need to go onto the “key” cell to open the gate, then the solution is even simpler (Turn right, Forward 4 cells, Turn right) allowing to propose a minimal beginning activity.

Fig 3B. The view of the conditional part of the path. A door can either obstruct the A gate or the B gate.
Thus, optionally, beyond a simple instruction sequence, a programming gate can randomly either open the A or B passage. If the A or B state is known before the robot starts, this only involves modifying the instruction sequence. If this state changes *after* the robot starts moving, this involves using a conditional expression, as shown on the right.

A step further, beyond the present activity, if the robot is hidden in the tunnel at the beginning, not only an ad-hoc algorithm, but a “generic” algorithm[19] has to be proposed that allows it to escape from any labyrinth

19 See, for instance, https://scratch.mit.edu/projects/69254134 for a minimal solution, of the form:

Repeat
  If in front of a wall
    Draw* a coin
    If the coin is on the head face
      Turn left
    Else
      Turn right
  Else
    Move forward
Until way-out

which is known to converge (very slowly but) almost surely to the way out, and has been observed to be easily inferred by beginners after proposing the simpler solution:

Repeat
  If in front of a wall
    Turn [always to the] right**

without knowing the labyrinth plan or the robot position. This allows the player to be left facing another level of abstraction.

The game is "always won", using the following cues:

- Some verbal explanation about the path to generate, i.e., key to open the gate, turn right to cross the tunnel.
- How to drive the robot to the key cell to open the gate, as a "starter", info solution is provided.
- A picture of the path under the tunnel, if blocked after the key cell has been reached.
- A verbal description of the whole solution to "translate" into code.
- A partial solution to complete.
- A last, the solution to reproduce by copy.
- Otherwise, by some "magic" the game is finally won.

Didactic objectives.

Here we target *COMPO3*, i.e., formal systems (algorithm) competencies, and *COMP5, i.e., program creation*, more precisely "*instruction sequence*" and "*parameter value*", while at a higher level "*conditional expression*" is also considered. The general programming paradigm is also experimented with here: It is to be broken down into 5 steps, (i) consider a problem, (ii) write a piece of code, (iii) execute it, (iv) analyse the result, (v) reconsider the problem. The key point is that these didactic ingredients are discovered and carried out to solve what can be a minimal simplest problem, and then reused to solve problems of increasing complexity.

Interesting enough, before playing with the robot on the tabletop, an unplugged activity called the "game of the robot"[20] and extensively studied (see, e.g. Romero, Duflot-Kremer, and Vieville 2019) when one player plays the robot while others "program" it in a 4x4 cells place corresponding to the game place, could be a fruitful way to start learning the know-how before transferring it to the tabletop situation.

Scientific culture.

Beyond these know-how skills (*savoir-faire*) the activity is also designed to share[21] some scientific knowledge (*savoir*).

On one hand, we can make explicit some knowledge tokens such as:

Else
Move forward
Until way-out

which often allows the robot to find a way out, unless trapped in a cavity. This can easily be simulated "under the tunnel", letting the robot run along a virtual (i.e., software calculated) path and then escaping in real life, when finding a way out in the virtual labyrinth.
A step further the Pledge algorithm https://interstices.info/lalgorithme-de-pledge (see also https://en.wikipedia.org/wiki/Maze_solving_algorithm#Pledge_algorithm) allows us to solve the problem in a much more efficient way, but with a much more complex method.
(*) Any random mechanism is suitable here, the only requirement is to *randomly* change from left to right, to never produce the same turn sequence that may correspond to some cyclic path.
(**) Indeed, it could also be proposed to always turn left, while hesitating between left or right is a lever to find out the idea to "change", i.e., sometimes to the left, sometimes to the right.
20Please consider https://youtu.be/9AtmJ9mTaB0 for a concrete reusable description of the robot game.
21To concretely illustrate how this is done concretely please consider:
+ https://project.inria.fr/classcode/profiter-de-classcode-en-postcast/ to see how abstract notion are linked to historical stories and fixed via everyday life anecdotes and
+ https://youtu.be/lHE-mT0l7pw?t=85 to see another sharing of Al-Kwarizmi story put in perspective.

-1- simple algorithms to solve a specific problem are made of a sequence of instructions;
-2- when problems correspond to variable situations, conditional expressions allow us to take these changes into account;
-3- some instructions are parameterized by numerical values, e.g., the sequence
move-forward-once move-forward-once move-forward-once,
is equivalent to
move-forward-3-times;
-4- instructions have some "algebraic" properties[22], i.e., can be simplified, e.g.,
turn-left turn-right,
or
turn-left turn-left turn-left turn-left,
is equivalent to ... do nothing, a kind of "zero" instruction, and so on; it is quite interesting to find such algebraic rules (e.g., turn-left twice is equivalent to turn-right twice) and find out a language to abstract these rules from what can be observed with the robot;
-5- with instruction sequence, conditional instruction, iterative instruction, and parameter value assignation, it seems that we can produce generic algorithms (we can produce all possible algorithms);
and so on.

On the other hand, it is also time to discover two great figures, back to 490-800 AD:
- Aryabhatta was an Indian mathematician and astronomer born in 476 AD. His only surviving work is called Aryabhatiya, a book that contains mathematical and astronomical theories, including how the decimal system works. His greatest contribution to mathematics and computer science was the invention of Zero.
- Al-Khwarizmi was an Iranian mathematician and astronomer born in 780 AD. He introduced Hindu-Arabic numerals in his famous book titled 'The Compendious Book on Calculation by Completion and Balancing'. He introduced the concepts of algebra into mathematics.
This double choice allows to show that computer science is not only a western and contemporary story[23], and also shows how science has been built thanks to intelligence sharing between civilizations. It also shows both the difference between these two formal sciences that are mathematics and computer science and their deep links.

Activity evaluation.

The activity evaluation is developed combining i) in-task observables, ii) debriefing questions to the player and iii) competence transfer activities.

In-task observables: At the present stage of our study we have identified the following pedagogical observable, which are:

- Time until checking the 1st solution and average time between executions.
    - If too long, some clues are proposed, including partial solution, as detailed previously.
- The number of executions of the program, before a solution is proposed.
- The number of times the instructions are viewed again.
- Level of clues given to find the solution.
- When a solution is provided, the edit distance[24] between one true and the proposed

22This is a very concrete example of the Integer modulo 4 commutative ring https://en.wikipedia.org/wiki/Ring_(mathematics)#Example:_Integers_modulo_4

23Contemporary scientists from non western countries are also quoted later, such as Maryam Mirzakhani, her work being of great importance for non-linear data representation and random algorithms, this to avoid the bias of "long ago, people from non western countries were brilliant" ;(.

24The edit distance https://en.wikipedia.org/wiki/Edit_distance here the Levenshtein distance is a string distance for measuring the difference between two sequences. Informally, it is the minimum number of single-item edits (insertions, deletions or substitutions) required to change one sequence into the other. It also quantifies the fact the player has found an approximate solution "closed" to a correct one.

solution.
- Identification of false but "clever" solutions (e.g., false solutions proposed by several persons).
- Robot moved with the end detection.
- This list will be completed after experimenting on a large scale with the first ensemble of players.

At the didactic level, observables are related to skills expected from the player, e.g.:
- Demonstrate a sequencing action
    - Observable: Placing the cards in the right sequence.
- Parameter value manipulation
    - Observable: Make use of the number of steps for the “forward card”
- Use a conditional expression
    - Observable: Make an appropriate test choice.
    - Observable: Identify the correct sequences as a function of the chosen test.

Beyond, if Ozon gets stuck, the player can analyze the reasons for this and try to fix the problem, by debugging.

Debriefing questions to the player: To evaluate the computational thinking beans beyond the activity, the following questions are proposed:
- How many steps for this algorithm?
- What is the path length as a number of squares?
- Are there several solutions possible? If yes, what is the optimal one?
- What happens if you quarter turn right four times?
- What happens if you quarter-turn left and right?
- What is the most difficult task?
- What was Al-Khwarizmi’s original name?
- What did Aryabhatta invented as a number?
- Given this hardware:
    - Can the A-closed and B-closed conditions occur simultaneously?
    - If the A-closed condition is true, what about the B condition?
    - Could[25] A-open and B-open occur simultaneously?

These questions are implemented via an online multi-choice quiz, with the possibility to add comments, to collect learning analytics, and the player can always return to setup and touch it. Then a “human” collective debriefing to discuss these issues would be of great benefit.

Competence transfer activities: The most sophisticated way to evaluate what has been learlearnednt is to propose some competence transfer activity, such as:
- Can program the same sequence in a Scratch environment[26].
- Can take other conditions into account.
- Can verbalize what is an algorithm at this simple level.
- Realize that true robots are much more complex but not qualitatively more intelligent than Ozon.
- Perform the activity of higher complexity, i.e., propose a general labyrinth way out algorithm.

At this level, we do not consider an automatic evaluation of competence transfer but plan to

25Both responses: no they can’t and yes they can ... could be ok ! It depends how we interpret the device, if we assume the mechanism has only two positions, they can’t, whereas if we consider an intermediate position (with the door juste between A and B gate) is possible, they can. It is very interesting to explain that both responses are ok, depending on what is hypothesized.
26The tool is available here https://scratch.mit.edu/projects/417551480/.

perform this manually with human interaction.

Activity 2: When cooking meets computing.

For the second activity, the idea is to formalize a cooking recipe as detailed in Fig.4.

Didactic objectives.

Here we target *COMPO3*, i.e., formal systems (modeling) competencies, and *COMP02*, i.e., problem modeling, while the didactic objectives are quite different here: The programming task itself is intentionally quite obvious, whereas the player is invited to discover and understand several notions beyond the usual programming.

The first notion is “modeling” which is the lever of computational thinking: How do we reformulate a “real-life” issue in such a way that it reduces to an algorithm applied on coded data? Considering the recipe example, what shall we neglect or forget to concentrate on what needs to be formalized for a naive- human or robot to succeed in doing the recipe. This is not simply a list of knowledge or how-to but a complex competence. Depending on the target performing the recipe, the modeling level differs: Some elements could be obvious or not (e.g., actions or gestures such as “pour” or “mix” may have to further decompose in more elementary instructions; objects like “dish” or “mold” may have to be identified by additional cues). In the present activity, a design choice has been made to offer a playful setup. A step ahead, the player is going to be questioned about this choice.

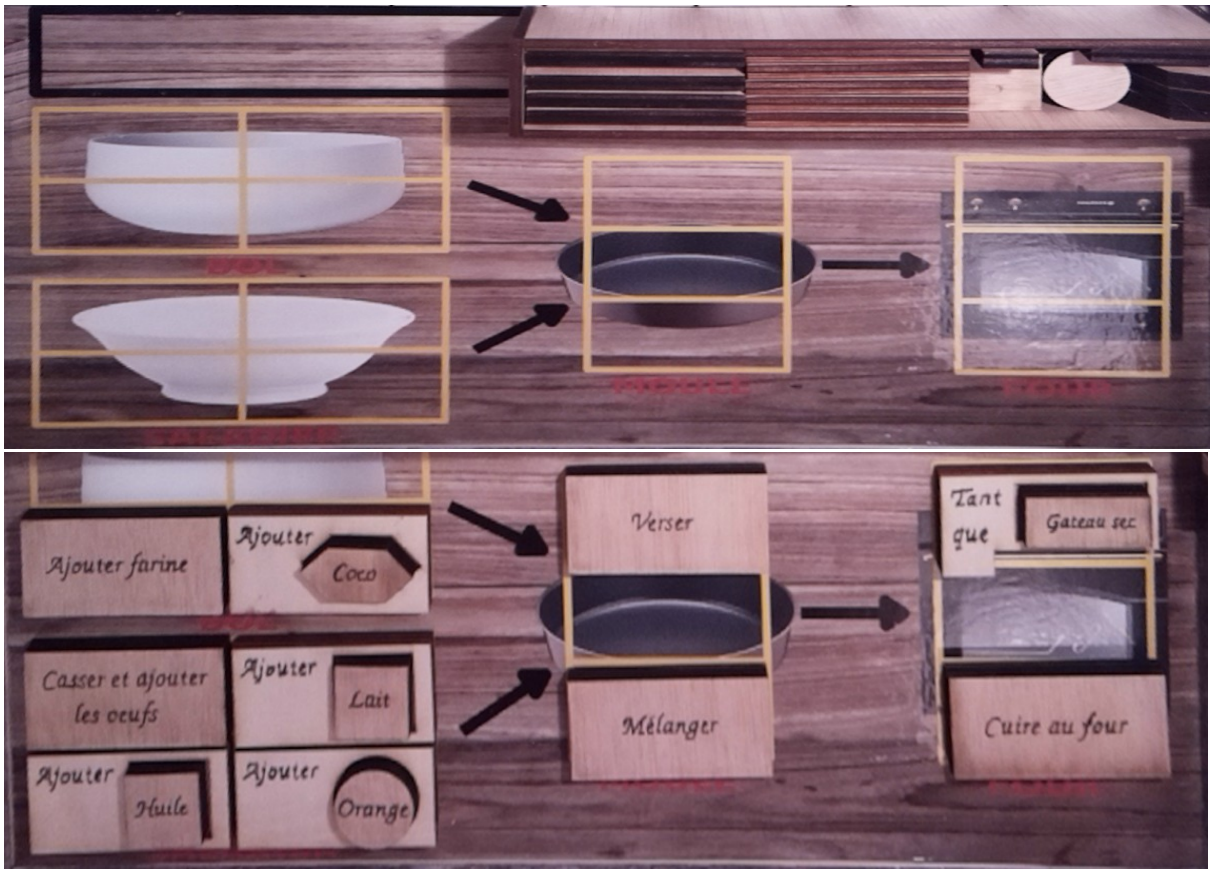


Fig. 4. A recipe is formalized, where dry components (flour, and sugared substances such as chocolate or sugar) are gathered in a dish and wet components (oil, milk, or water, fruit essence and flavours, eggs) gathered in a bowl, before being poured and mixed in a mold and baked in an oven. This last operation requires a loop, with the right condition (i.e., while the cake is soft or wet, do bake). The top view shows the “programming area”. The bottom view shows wood cards for a partially realized recipe. The player is invited to specify a recipe, for a gnome that is so stupid that we must provide each action with the proper ingredient very precisely, this is

explained on the panel so that the gnome can execute them correctly. Any solution is acceptable, providing that some constraints are taken into account, for instance: Using flour and a sugary substance, eggs, a liquid and any essence. Beyond this intentionally simple activity, several computing concepts are introduced, as detailed in the text.

Furthermore, several notions beyond the usual basic programming are invoked here:
- V*ariables have a type*, i.e., their value is to be taken in a given set or domain (here some values are liquid, others are sugared, etc). This is a concrete example allowing to understand that a "value" is meaningless unless its domain is specified and for numerical values, also the used unit (e.g., is the milk quantity given in terms of weight (in kilograms or English (or other countries?) pounds?) or volume (in litre or gallon or other?)).
- *Feedback loops allow autocalibration*, i.e., automatically adapt a process to an unknown or unstable parameter value (e.g., instead of baking the cake 10 or 20 or ... minutes, which depends on the oven, the cake weight, etc, we loop on baking it a few minutes until it is dry, the unknown baking time being also measured and known at the process end). This allows us to understand
- *Some operations are parallelizable*, i.e., can be performed in any order, thus in parallel (e.g., gathering the elements), while others must be done in sequence (e.g., mix after pour). This notion is related to the *dependency graph*, a directed graph representing dependencies of several object productions towards each other.
- *Programmation can be defined "by constraint"*, i.e., the algorithm is not explicitly specified by a sequence of operations but by the desired outcome and specified constraints (e.g., what kind of ingredients, which possible essence). Then the operation sequence is derived manually (or automatically).

The game is "always won", using the following cues:
- Some additional verbal explanation[27] about the recipe,
    - re-explaining that wet/dry substances gather in the bowl/dish,
    - making explicit whether each substance is wet (e.g., eggs) or dry (e.g., sugar).
- Solution variants are accepted as valid, such as
    - mixing the ingredients in the bowl and dish before pouring then in the mold,
    - not making explicit pouring since it is obvious.
- Missing operations are gently reminded, with increasing details, e.g.,
    - "what about mixing the ingredients in the mold?"
    - "didn't you forget something?"
        - "didn't you forget the essence?"
            - "didn't you forget the orange essence?"
- A partial solution to complete.
- A last, the solution to reproduce by copy.
- Otherwise, by "chance" the gnome will finally do it.

Scientific culture.

27Some supplementary hints are:
- Each ingredient has a special form to fit on the block.
- Wet ingredients must be put together in the bowl.
- Egg and oil, milk or water are wet and must be put in the bowl.
- Dry ingredients must be put together in the dish.
- Flower, sugar, coconut or chocolate are dry and must be in the dish.
- We must cook while the cake is weak.

While we may also notice, that in real-life additional constraints have to be taken into account such as:
- Egg must be broken before adding it.
- Fruits must be peeled before adding them.

These show that the present game is a simplified model of reality.

Previous modeling issues and related notions are carried out implicitly during the activity. The overview time allows the player to make these notions explicit.
The first pedagogical lever is to develop *inductive abstraction by examples*. For instance, we notice the particular fact that ingredients gathering[28] order is not important, the action being parallelizable, and then invite to infer that for several other situations the operation order might (or not) be important, e.g., addition of numbers, for instance, "1+3+2+4" can be calculated in any order and the first two additions can be done in parallel (1+3 →4 in parallel to 2+4 →6 and then 4 + 6 → 10). Understanding what is parallelizable or not helps for instance, organizing human teamwork.
The second pedagogical lever is to develop *generalization by variable values*. Very simply, if you know how to cook a chocolate pound cake, you immediately know how to cook any, say, orange pound cake or another vanilla and nuts pound cake, providing you create a variable "essence" and adjust its value. This capability to adapt a given way of doing, here a process, by parameterization is another quite useful computational thinking skill.

On the other hand, it is also time to discover two great figures of the 19th century, one century before the computer age.
- Charles Babbage was a mathematician, philosopher and inventor born in 1791 in the UK. He is often known as the 'Father of Computers' because he provided detailed plans for mechanical Calculating Engines, Difference Engines, and Analytical Engines, which appears to be the first computer.
- Ada Lovelace[29], born in 1815 in the UK, is known as the first +computer programmer. She translated an Italian article on Babbage's Analytical Engines, and added her notes on how codes could be created for a device to handle letters and symbols along with numbers. She also theorized a method for the engine to repeat a series of instructions (known today as looping) that computer programs use today.

Activity evaluation.

In-task observables: At the present stage of our study we have identified the following pedagogical observables, which are, as for activity 1, time until checking the first solution and average time between trials, number of trials before finding a correct solution, the distance between the proposed solution and a correct one. In addition, specific observables are identified:
- originality of the solution (for solutions proposed by other persons)
- predefined errors identification (inversion of blocks, improper choice of location, block forgotten).

At the didactic level, observables are related to skills expected from the player, e.g.:

- Understand the ingredient type constraints.
    - Observable: Properly associate ingredients and action parameters.
- Understand the action causality.
    - Observable: Put related actions in the correct order.
- Understand the wet/dry ingredient parallel treatment.
    - Observable: Put ingredients together correctly, put them in the right container.
- Understand the cooking loop.
    - Observable: Introduce the cooking in the loop and choose the correct condition.
- Understand the difference and importance of sequential and parallel steps in a process
    - Observable are those of items 2 and 3.
- Realize that the same plan recipe or plan, can produce various results depending on the variable values.
    - Observable: Redo a variant of the recipe by simply changing some ingredients.

28Cooking chief might counter-argue that it might not be exactly the same to gather flour before, say, sugar that the reverse, here by separating dry and wet elements before mixing, we are on the safe side, regarding usual cooking at home.
29See https://youtu.be/lHE-mT0l7pw?t=149 for an example of a video presentation of this scientist.

Debriefing questions to the player: To evaluate the computational thinking beans beyond the activity, the following questions are proposed:

- How many operations to do the recipe?
- If two gnomes make the recipe together how to share the operations?
    - Which one can not be shared?
- Give an example of two operations you can invert, and two you can not.
- If you redo a coconut strawberry cake instead of this one, do you need to redesign everything?
    - If not what do you change?
- Considering the different possible ingredients in the box, how many kinds of cake can the gnome make?
- Are “lemon” and “pear” of the same data type? And what about “sugar” and “milk”?
- What is the type of elements represented by a circular chip?
- How to know the best cooking time using the proposed loop?
- What was the main scientific contribution of Ada Lovelace?
- What did Charles Babbage plan to do?

Competence transfer activities: The most sophisticated way to evaluate what has been learned is to propose some competence transfer activity, such as understanding what is a variable is a skill

- Washing machine modeling ;) for instance:
    - the washing machine temperature is a variable,
        - allowing to wash different kind of clothes,
        - using the same washing sequence, but the water temperature,
        - physical reasons to adjust the water temperature could be challenged;
    - there is another quantitative variable (the spin-drying speed);
    - there are also qualitative variables (e.g., pre-washing cycle or not) ;
    - not only women can use such a machine, whatever the complexity is, because, with computational thinking formation, even men would likely be able to.
- In Scratch variable notions is also widely used at a different level[30]:
    - modify a parameter (e.g, move 50 and not 30 pixels),
    - creates a variable (e.g. the speed of the sprite) and set/get it,
    - understand the difference between quantitative (e.g. turns 85, degrees left) and qualitative (e.g,. quarter-turn, u-turn, ...),
    - uses this variable in a process,
    - design a “function”. i.e. a process that is parameterized by some input variable value.

As for the previous activity, in-task observables correspond to online measures, debriefing questions to post-test, and competence transfer is related to learning sequences beyond the present study.

Activity 3: playing with pixels to twig coding.

This activity, described in Fig.5., is not related to algorithms and programming but to the other main chapter of informatics initiation, namely information representation and data coding. The activity is usually performed[31] between two players, while only a reduced form is considered here, since we have to measure learning analytics, and want to focus on the didactic aspects of the activity.

When performed between two players, the instructions are minimal for the two players to solve the

30This is illustrated with several examples here: https://scratch.mit.edu/studios/27331752/
31See http://tinyurl.com/y8btzny7 for a description.

"transmission" problem: Here, «one player draws a pixel art without showing it, and can only say any number of "0" or "1" to transmit it to the other player that has to redraw it». Both players have to discuss together how to do it. They have to agree on how to code (i.e., the "0" and "1" pixel correspondence, the pixel encoding sequencing, not necessarily from left to right and top to bottom). This is a very interesting problem-solving situation.
Sometimes, the players "cheat" in some very interesting manner to encourage: instead of repeating, e.g., "00001" for instance, they say "four 0, 1", enriching the initial coding language. When this happens, it is an opportunity to notice that it was not the initial rule, but an improved one. Sometimes, the players also develop some protocol to ensure that the code is properly transferred, even in noisy environments: Repeating the sequence, or feedbacking what has been received by the receiver, for the emitter to check. Another example is two kids who wanted to transmit not only two color pixel art, but multicolor, thus invented a specific code to this end.
Furthermore, errors are very interesting situations, it is first an opportunity to remind about the pedagogical interest of making errors: «who makes no errors learns nothing new, but simply remains on what is learned before»; an error is a feedback[32] that allows her or him to adjust his or her behavior to improve it. Some errors are demonstrative, if, for instance, the pixels are drawn from right to left instead of left to write, the result corresponds to the original picture via a reflectional symmetry. If the result was a '5' it will look like a '2', if the result was a '0' it will look like itself, providing an interesting way to make concrete geometry.

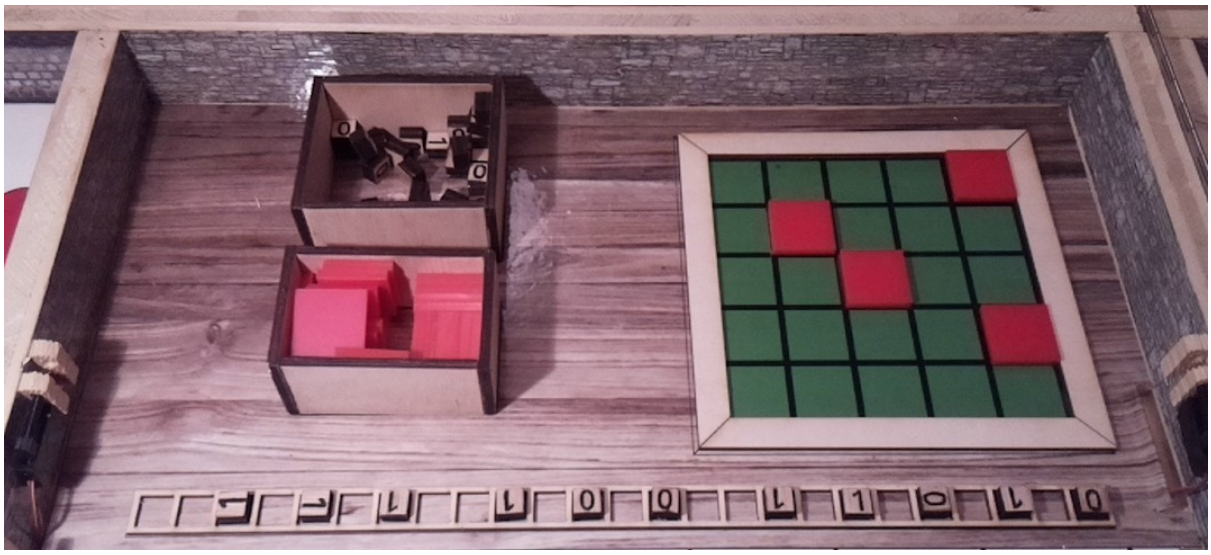


Fig.5. A 5 x 5 pixel art picture can be built to be binary encoded, for instance: 0001 01000 00100 00001 00000 (green squares are encoded with '0' and orange squares with '1'), corresponds to our view of the picture visible here, reading from left to right and top to bottom. The decoding exercise consists of considering a binary sequence, say, 011101010110101011101 1011 (which corresponds to a kind of imp), and to draw the corresponding pixel art picture. The encoding exercise consists of drawing a pixel art picture of your choice and then binary encodes it.

Didactic objectives.

We target *COMPO3* formal systems (information coding) competencies here, including coding and decoding of information, image pixel, binary code, transmission of information, and to some extent *COMP05*, i.e., creation (not of a program but oa digital object).
The basic notions to be understood regarding information coding in link with this activity are the following:
- *Atom of information*: Any couple of values (e.g., "0"/"1" or "yes"/"no" or "true"/"false") allows to define the minimal information token, i.e., an atom of information. This can be experimented with

32This is also entirely true and completely formalized in machine algorithmic learning, based on supervised learning examples, or reinforcement learning reward.

by playing the well-known “guess who?” game (as an example), which also provides the opportunity to understand the dichotomy[33].
- *Information additivity*: Two disjoint, i.e., non-redundant information add, e.g., being a human, and being old, contrary to induced information, e.g., being a boy, and being a human. This has a probabilistic interpretation (the probability of the union of two disjoint events add) but is also true at the computational level, for instance, considering the number of bits to encode it.
- *Arbitrariness of coding*: It is entirely equivalent to encode, say, green or orange colors with “0” and “1” or vice-versa, the only important point is that we *agree* on the chosen standard. A lot of technical words such as “.mp3” or “.jpeg” named standardized formats to encode sound or images. The UTF-8 standard, for instance, encodes almost 250000 letters and characters of any human language.
- *Information weight and size*: The “raw” information weight is the number of bits to encode the related data, since the data is often redundant the information size is often lower, as it can be measured for instance, using a zip lossless compression algorithm, while other compressions (such as those used in “.mp3” or “.jpeg” formats are lossy, neglecting information that is not or dimly perceived by the human auditory or visual system).

Scientific culture.

On one hand, beyond the previous didactic objectives, the player is invited to realize all information is coded in binary, including, say, emotions[34], understanding that coding the emotion in the machine, and even simulating them, does not mean that the machine *feels* the emotion.
The fact that all human information can be coded as a sequence of binary bits yields a revolution. In a nutshell:

- Generic mechanisms to memorize, transmit, compress, crypt, information can now be applied to text, images, sounds, videos, and any data. This is also the case for machine learning mechanisms, which qualitatively change what can be done with our information. This has also negative consequences, such as data perennity.
- Because all information does not need specific support (e.g., an image does not need to be stored on a silver halide photograph, music on a vinyl record or magnetic tape) the copy cost becomes negligible and is no more a rival good[35]. This completely changes economical models and has a huge environmental impact.
- For all encoded information to be reusable and interoperable, it must be standard compliant. All countries in the world, including countries at war, have accepted to follow informatics standards. Some companies' commercial policies, like Apple or Microsoft, have been based on “proprietary format” so that customer data was “trapped” on their systems; it was finally not tenable and open format is now widely used.

Understanding how information is coded in digital systems allows us to better understand these disruptive effects of informatics on many societal aspects. Interestingly, scientific culture is here multidisciplinary linking computer science concepts to human and social sciences.

On the other hand, it is also time to discover two great figures[36] of the 20th century, at the origin of the computer age.

33Dichotomic search is a general algorithmic principle allowing to optimally search an element in a sorted set.
34Considering, e,g., the Plutchik 3D model of emotions https://en.wikipedia.org/wiki/File:Plutchik-wheel.svg as explained here https://en.wikipedia.org/wiki/Robert_Plutchik#Plutchik's_wheel_of_emotions, the 32 emotions considered as discrete categories (which is a modeling choice), can be encoded with 5 bits, 2 coding the emotion intensity, 1 the emotion antagonists, 2 coding the emotion basic type.
35A rival good https://en.wikipedia.org/wiki/Rivalry_(economics) consumption by one consumer prevents simultaneous consumption by other consumers, which is not the case for a joke ;) or a digital object.
36See https://youtu.be/3YrmbBsh2Ig for a presentation of both characters, in their historical context, and https://tinyurl.com/yd3h99sk to see how the link between these persons and computer science concepts are proposed.

- Alan Turing was a British scientist and pioneer in computer science born in 1912. During World War II, he developed the Turing Machine which helped break the German Enigma code. He mainly understood what a machine can execute and what it can not, showing the limits of what can be calculated by an algorithm. Even so, he believed in artificial intelligence.
- Grace Hopper was born in 1906 in the USA. During World War II, she worked in computing for the US Navy. After the war, she led a team that created the first computer language compiler, which led to the creation of the popular COBOL (stands for Common Business-Oriented Language - a compiled computer programming language similar to English) language.

Activity evaluation.

In-task observables: At the present stage of our study we have identified the same pedagogical observable as for the previous activities.

At the didactic level, observables are related to skills expected from the player, e.g.:

- Create an image using 0 and 1, and make her/his choice.
    - Observable: Being able to create a pixel art of her/his choice without blocking on the "but what do I have to work.
- Understand how pixels work.
    - Observable: Being able to create a 5 digit by putting squares on the grid (any form that is like a 5 is ok).
- Understand that a picture with pixels can be encoded with 0 and 1.
    - Observable: Ability to redraw the pixel art from a 0 / 1 sequence.
- Make the competence transfer to encode an existing drawing.
    - Observable: Do the coat of arms proper encoding.

Debriefing questions to the player: To evaluate the computational thinking beans beyond the activity, the following questions are proposed:

- If I draw an S, but encode it from right to left, what symbol will it look lik?
- Suppose each pixel can be red, blue or magenta (red + blue mixed): How many bits (i.e. 0/1) do we need for each pixel?
- If I made a mistake between 0 and 1, what happens?
- If I have a 5 x 4 pixel art grid, how many 0/1 do I need to encode it?
- If I encode the grid from right to left and others who decode also consider right to left, is the picture going to be correct?
- The main contribution of Grace Hopper was the creation of a compiler. It translates a human-readable computer language into ... what?
    - [another human language,
      binary code that the machine can execute,
      a secret code that no machine can decode]
- The main Alan Turing contribution was to understand what a machine can calculate, one of these assertions is wrong, which one?
    - [a computer machine can execute only algorithms
      a computer machine can execute all possible algorithms
      Some problems can not be solved by algorithms
      any problems of any kind can be solved by algorithms]

Competence transfer activities: The most sophisticated way to evaluate what has been learned is to propose some competence transfer activity, such as:

- Pixel encoding on other situations, e.g.:
    - Draw a pixel art using another mechanism, e.g., clicking on the screen, as proposed here https://scratch.mit.edu/projects/69242142
    - Use pixel art drawing to observe some phenomenon such as random generation, as proposed here https://scratch.mit.edu/projects/287569868

    - Apply pixel encoding/decoding on
        - other pixel art array (e.g., 4 x 5) or
        - using another binary vocabulary (e.g. O||O|OO…or YNNYYNNY… (for yes and no)) or
        - another encoding, e.g.,
            - three bits Red Green Blue to generate 8 colors
            - with the possibility to insert digits for bit repetition (e.g. 0111001 becomes 031201)
- Encode not a pixel art but another digital object, e.g.,
    - the activity 1 algorithmic sequence using F L R (for forward, left or right)
    - a multi-drump sequence,
    - a navigation North/West/South/East navigation algorithm.

As for the previous activities, in-task observables correspond to online measures, debriefing questions to post-test, and competence transfer to related to learning sequences beyond the present study.

Activity 4: the crea'cube problem-solving test.

The last activity consists of « building a vehicle made up of four pieces that moves by itself from the red point to the black point » (instructions can be heard again by pressing the button), as shown in Fig. 6. It has been extensively detailed in (Romero, David, and Lille 2018), thus not redeveloped here.

Didactic objectives.

The following CT competencies are targeted:

- Analyse a problem situation and imagine the way a set of 4 cubes can be used to solve the challenge (COMPO1) through the creation of an autonomous train (COMPO2).
- Understand the importance of the order in a sequence (system behavior defined by the order of the different cubes) (COMPO3)
- Understand basic physical systems components (COMPO4): cubes assembled as a system, distance sensor, movement actuator (servo-motor + wheels), electric circuit
- Engage in computational thinking as a problem-solving activity requiring the creation and test of a solution (COMPO5) and the iterative improvement of it based on the problems observed (COMPO6)

As illustrated in Fig. 7.

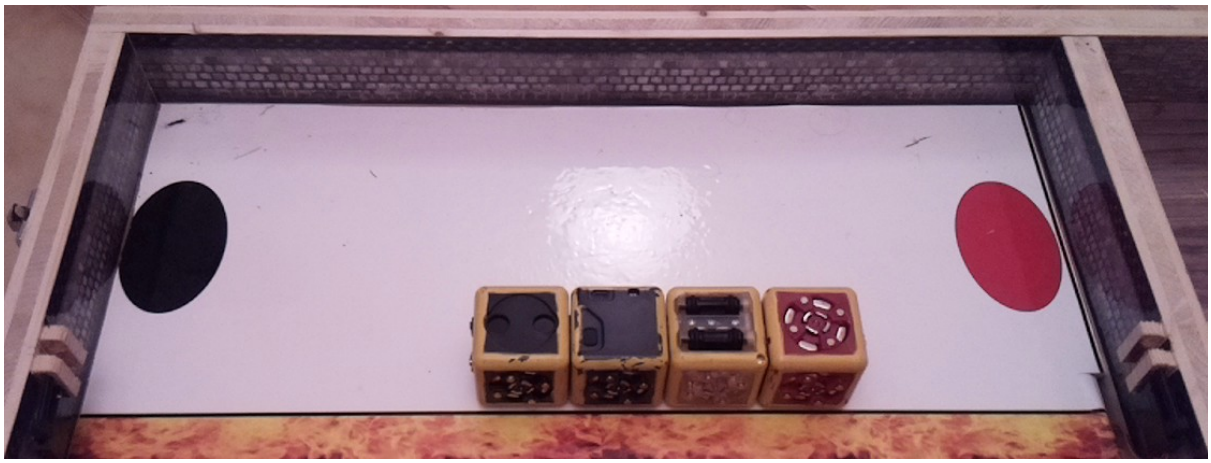


Fig. 6. Four cubelets have to be assembled to form a vehicle that moves from the red to the black blob. This is an open, voluntarily ill-defined, task allowing the player to develop problem-solving and creativity skills.

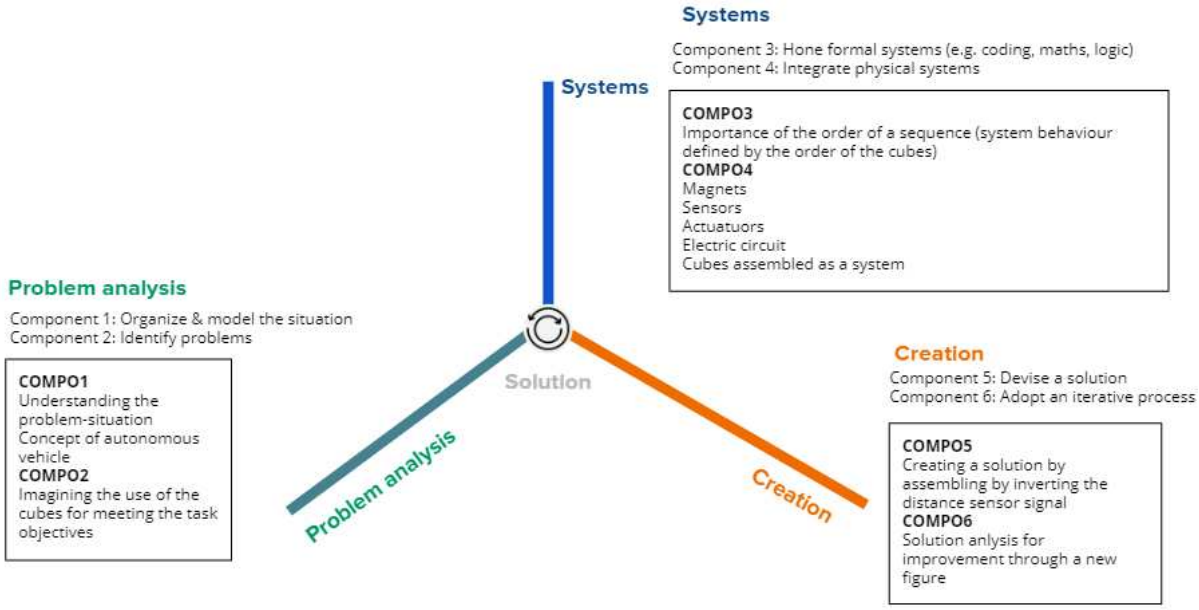


Fig. 7. Targeted competencies in relation to the computational thinking framework as developed in (Romero, Lepage, and Lille 2017).

More details are available in (Romero, David, and Lille 2018).

Scientific culture.

On one hand, the problem-solving situation is a great opportunity to wonder how a machine would *learn* to solve such a complex task, i.e., to compare natural and artificial intelligence[37] (Alexandre et al. 2020). Thinking about how the task could be solved "mechanically" is a very interesting way of thinking in a reflective way how a human brain can solve it *systematically*.

Learning to verbalize mental operation, is a key of what is called metacognition (Lachaux n.d.) (Diamond and Ling 2016), and it is proposed here to make a profit of this particular setup to not only perform the activity as a subject but use this activity to initiate a mechanism of learning to learn.

37A large audience citizen formation https://classcode.fr/iai has been proposed and widely shared with a focus on the links between artificial and natural intelligence as quoted here,

On the other hand, it is also time to discover two great figures of the early 21st century, Yann André LeCun is a French computer scientist working primarily in the fields of machine learning, computer vision, mobile robotics, and computational neuroscience. He is well known for his work on optical character recognition and computer vision using convolutional neural networks (CNN), and one of the fathers of deep learning, i.e., numerical artificial intelligence.

Rose Dieng Kuntz was a Senegalese computer scientist born in 1956. Her research primarily focused on sharing knowledge over the World Wide Web, and she specialized in semantic web models of knowledge, i.e., symbolic artificial intelligence.

Activity evaluation.

In-task observables: The observables related to this activity have been detailed and discussed in (Romero et al 2020), and are not reproduced here, but reviewed in Fig. 8. However, in the present implementation of the activity, we introduce a new paradigm regarding this activity evaluation: monitor[38] the activity by the previous player.

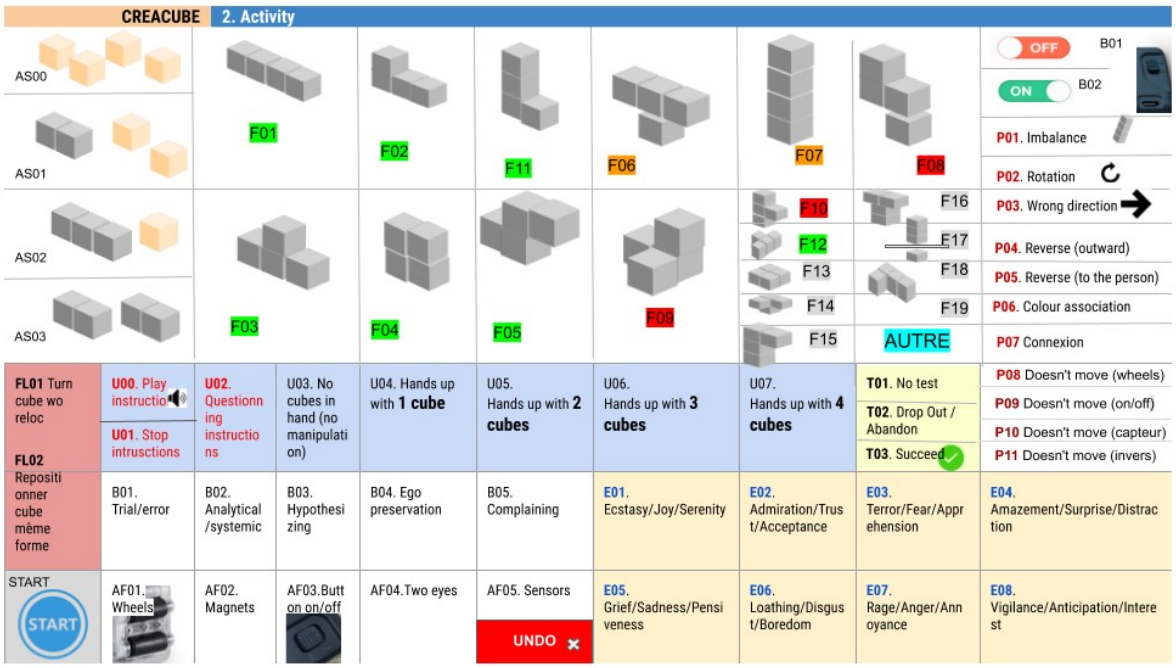


Fig.8. Observable of the crea'cube activity: the cube complete configurations F** or partial assembly AS**, particular usage U**, general behavior B**, observed emotion E**, discovered affordances AF**, generated production P** and terminal state T** correspond to the discrete categorization of the experimental states. They are recorded with the precise instant of occurrence. They are measured offline on a video recording of the experiment. In the present modified paradigm, each configuration is presented on the tabletop, and either a human observer or a machine learning algorithm is identified in real-time in the experimental state.

The idea is to invite a player to master the next player activity, after the debriefing session, proposed below. This not only allows to save experimenter time but is a real lever for the former player to better understand how she or he performed the problem-solving task, i.e., it is an interesting way to learn how to learn.

38This idea has been introduced because automatic recognition of the 3D configuration is not fully operational, however, due to the intrinsic interest of this fallback solution, we maintain it even if the implemented machine learning mechanism is efficient enough.

The role of the former player is simply to passively record in real-time the different configurations, and other elements of the experiment as shown in Fig. 8.
Of course, an experimenter must supervise the binome, to check that the former player does not simply provide the solution (because of, e.g., lack of patience) or disturb the player (because, e.g., envious of a better efficacy), and previous data collected after a passive video recording will allow us to observe to which extent this introduces a bias at the player level.

Debriefing questions to the player: To auto-evaluate the computational thinking skills summoned during the activity, the following questions are proposed:

- Did you, the player, find the solution by chance?
- Which cues did help understanding how to assemble the cubelets?
- What was the "spurs" (i.e., the critical cue) that allowed you, the player, to find out one aspect of the solution?
- According to the experiment state definition (shown here in Fig.8), how would qualify your emotion at the beginning, middle and end of your session?
- Looking at the collected learning analytics, to which extent do you agree or what would you propose to correct?
- How would a player advise another (without providing the solution) if stuck at the beginning of the activity?

Competence transfer activities: This aspect is not treated and is an interesting perspective of the present work.

Designing a modular automatically monitored tabletop

Position of the problem

We are in front of the following challenge: Design a tangible device, low-cost and as much as possible using recycling material, for which the construction and following upkeep can be done by makers with a relatively low level of technicity. Furthermore, we want teachers (in the wide sense) to be able to derive or redesign their activity, possibly not only in relation to computational thinking initiation.

To take up the challenge we have considered both open-hardware and open-software productions. The key point is that the method allows a low-cost reproduction of the tabletop or any variant, and is easily reusable for other setups.

Open-hardware production

The present mechanical realization is made of standard woodworking, using recycled wood, it is nothing more than a crate with a double bottom, the electronic being in the cellar and the gameplay space above.
Regarding the object animation, low-cost electronics has been considered, with standard angular and linear servo-motors and a low-cost EPS32 controller, as sketched out in Fig. 9. This last choice (for instance, Arduino) allowed a lower cost, better performance and easy control via a wifi web-service interface. This component is used in connected objects. It is not obvious to make use of it, but it is now shared with a complete interface software which makes it completely obvious to integrate without any code development.

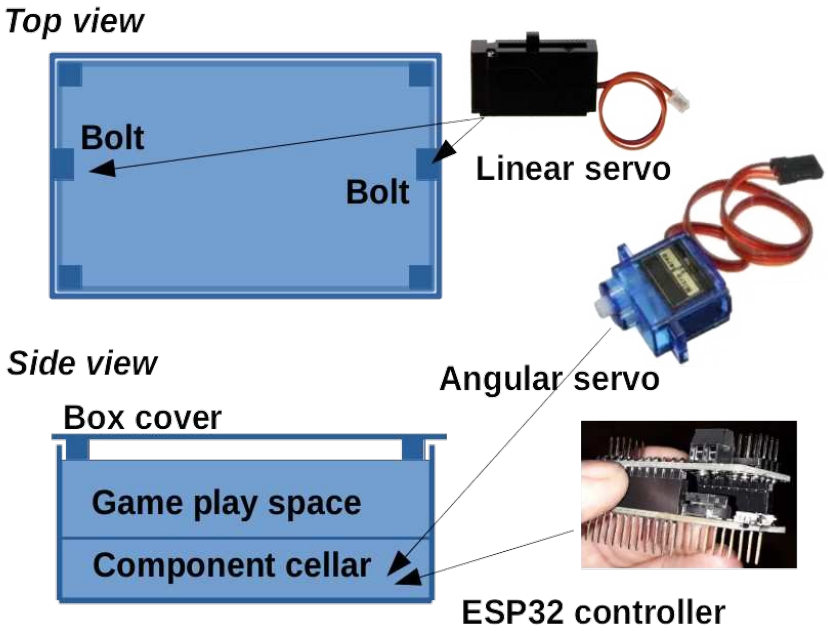


Fig. 9. A schematic description of the hardware design choices, using low-cost easily reusable hardware components, see text for details.

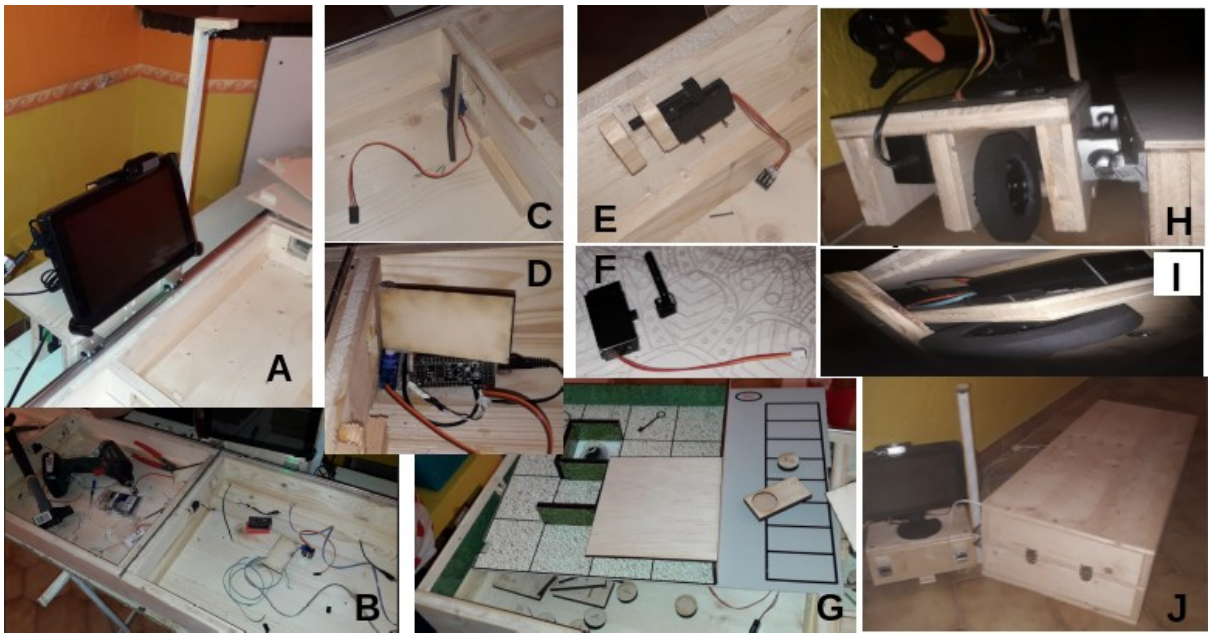


Fig. 10. Hardware photo gallery.
A: The Raspberry computer tablet computer mounted on a carriage with the image analysis camera Raspicam mounted on a pole to provide a top view of the gameplay space.
B: A view of the worksite, showing that everything has been built with standard craft tools.
C and D: Detailed views of an automatic gate between two rooms, with the controller stick in the cellar part of the box.
E and F: Detailed views of the locker mechanism, the bolt fastener has been produced on a 3D printer in a maker-space.
G: A view of the game board before putting it in place, all decoration is printed on a contact paper stuck on a thin wooden plank.
H and I: Detailed view of the wagon mechanism allowing the tablet to move along the game sequence.
J: A view of the setup when closed, allowing to easily move it from one place to another.

The controllers drive angular servo motors that open or close gates during the activity and linear actuators that allow to lock or unlock the bolts.
Each box, corresponding to each activity room, is initially locked with two bolts, as shown in Fig,9. This latter function works fine but is not the cheapest one and seems fragile: We advise implementing another solution in the next version, e.g., a number padlock which combination is revealed when it is time to open it.
A step further, as visible in Fig.11 and Fig.10, the tablet is mounted on a wagon that moves along a rail to automatically position the computer and top-view camera on the right position. This works, but is the second fragile part of the system, and cost may be reduced by simply considering four fixed cameras and manual manipulation of the tablet.
The main component is a Raspberry computer, installed at a low-cost (<200€) SunFounder RasPad tablet with a Raspicam camera mounted on a pole to provide a top-view of the gameplay space, as shown in Fig. 10. It also has a control card to drive the continuous motors of the wagon.
All the gameplay board and decoration is simply made out of contact papers stuck on thin wooden planks or on the box walls, which appears to be a rather low-cost solution contacting a printing shop, for which it is a very standard service.

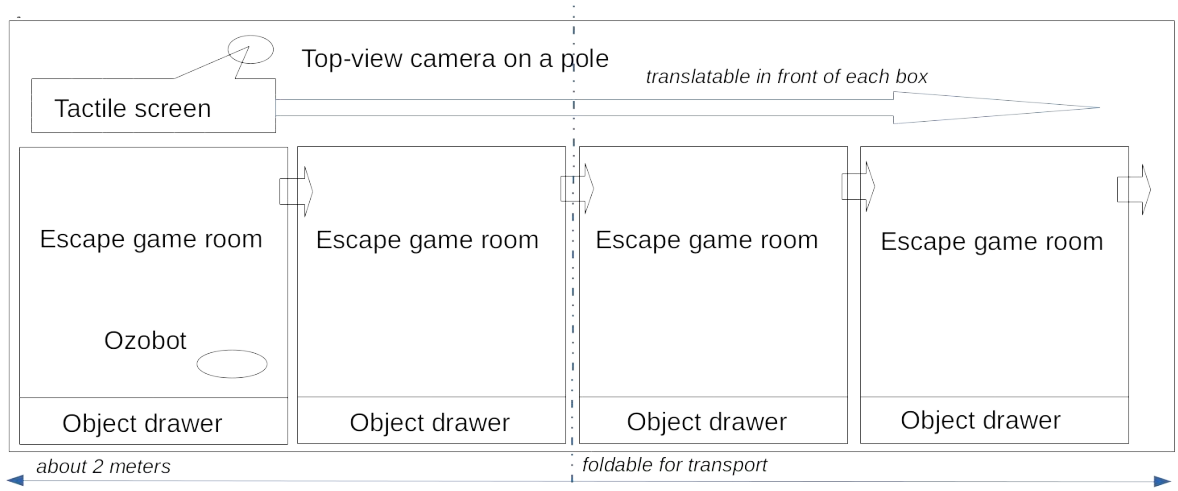


Fig. 11. Global top view of the present realization showing how the game runs from one escape room to another, each cover being unlocked when the previous activity is done, while the tablet mounted on a wagon with the top-view camera automatically follows the player.

This not only provides a rather low-cost solution but also a quite nice result for a rather small amount of work time. Beyond a non-negligible investment for finding this solution, producing the whole hardware is a one day work to buy the wood and electronic components (the latter being all available online with easy delivery), one day to build the wood structure and one day to mount the inside part of the setup.

As being an open hardware production, all files[39] (general plans with dimensions, list of components with (non-exhaustive !) link to buy them, files to produce the 3D elements and contact paper).

The level of competence for the woodwork is that of a handywoman or handyman, using home craft tools. The level of competence for the electronic is of the same level, some welding and wiring are to be realized. The best solution is perhaps to join a maker space or a fab lab and perform the realization in this context. This is what we did.

Open-software production

39Refer to https://gitlab.inria.fr/line/aide-group/aide/blob/master/etc/install_hardware.md for details.

To offer the most flexible and easy to use, develop and derive software, we have designed a relatively general architecture, detailed in Fig. 12. At the outset, the architecture is made of three elements: (i) The experiment setup, (ii) The data analysis machine and (iii) the software development repository.

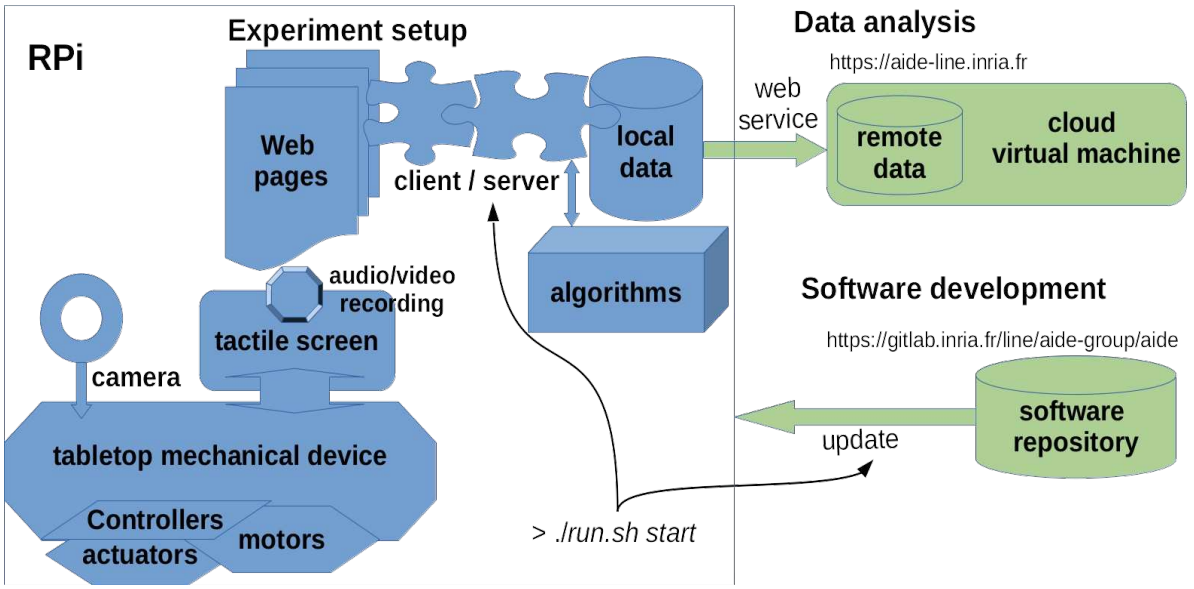


Fig.12. The system architecture. In blue are the Raspberry (RPi) components that run the experiment setup: drivers of the different hardware components, web pages to interact with the player, algorithms and local data storage. This section is connected via a local secure wifi connection to any desktop computer. In green the two external machines, one is a virtual secured machine to store all data and perform data analysis, the other is the software development repository. See text for more details.

To understand in-depth how this works, let us detail the different elements.

General software features. The AIDE software[40] bundle is defined as a set of factories, i.e., a set of simple static functions that encapsulate[41] all the implementation complexity and classes, i.e., digital objects (i.e., an Image or the Tabletop interface) with parameters to define their properties and methods to activate some functionalities.
It is shared under the CeCILL-C (a BSD-line license compatible with French regulation rules) for the code and CC-BY licenses for the other resources.
The core is programmed in C/C++ for both efficiency and capability to interact with other programming languages, mainly:

- Python for which all classes and factories are wrapped allowing to write code with this language (not often used).
- Javascript for which all needed functions are wrapped[42] and used via a string interface.

The applicative layers and user interface are designed in HTML5/CSS and programmed in JavaScript. This choice has several advantages: It is easy to create since many of us know how to create web pages, users are familiar with Web pages interactions, browsers environment offers many multimedia features, it runs everywhere (e.g., it can be developed on a standard computer before

40Available at https://gitlab.inria.fr/line/aide-group/aide/-/blob/master/README.md and documented at http://aide-line.inria.fr.
41The http://aide-line.inria.fr/build/www/sys.html lists all used system functionalities, lightning the use of the middleware underneath.
42See http://aide-line.inria.fr/build/www/wrapperService.html for wrapped functions details.

running on the setup, or an internet demo version can be shared), it can be used in remote configurations (i.e., setup driven by a host computer), it allows to produce hybrid (unplugged and online) activities. A toolbox is provided to ease the use of this design choice, as explained next.

To manipulate symbolic information we make intensive use of generic data structure, for both defining the parameters or obtaining the processing results. Very simply we can define, t-uples (also called table, or dictionary or associative array) of values of the form

{ name: value other-name: next-value ...},

Corresponding, for instance, to web-service URL query, and list of the form

[first-value second-value, ...],

each value being itself a t-uple, a list, a textual string, a number or a boolean (i.e., with either true or false value). This corresponds to a weak syntax JSON[43] data structure. We provide such detail here, because each user (from developers to data analyser) has to understand this simple knowledge representation.

Local server/client mechanisms. A second design choice is to consider the user interface via Web pages, as discussed previously, with the difficulty that a Web page is (hopefully) not allowed to freely interact with the host computer, for obvious security reasons. Here to manage this issue, we designed a minimal client/server mechanism, both sides written in JavaScript to limit the amount of technology, the server-side using node.js middleware. The service provides the capability to have remnant data local service (i.e., save and load data structures), and drive all tabletop hardware and software functions, while wrapping functions are easily extended. A very simple interface[44] is designed encapsulating all required elements.

Hardware components drivers. Actuators (linear or angular servo-motors) or motors (continuous servo-motors) are driven either by ESP32 controllers, via a wifi web-service, or the RPi servo card, via the Python Adafruit standard library. The Ozobot is connected to the infrared interface and also driven via a Python middleware. The other devices (camera, audio and video input and output, tactile screen) are driven by the RPi standard drivers or middleware. This rather complex mechanism is entirely encapsulated in the tabletop low-level driver factory[45], that takes all underlying tasks (e.g., initialization, finalization, error detection) into account. This low-level interface has been designed to be entirely reusable.

At a higher level, all commands of the present setup are encapsulated in simple commands to open or close gates, lock or unlock covers, and so on, for instance:

```
tabletop.wagonMove(distance);
```

to move the wagon supporting the tablet a certain calibrated distance.

Image processing implementation. The image processing is based on the https://opencv.org middleware, reusing all available functionalities with two add-ons:

- Fast implementation in C/C++ of the application-specific algorithms or combinations of existing algorithms to maximize the performances.
- A high-level interface to use these functions without directly manipulating the non-trivial openCV data structures.

An image[46] is made of a `photo`, which is assigned from some source (e.g., a camera, a file, an assignment from another image), with additional channels calculated by the implemented algorithms (e.g., color hue, thresholding, ...) and providing parameters extracted from the image

43The JSON data structure https://en.wikipedia.org/wiki/JSON requires a rather strong heavy syntax, while a weak syntax parser http://aide-line.inria.fr/build/www/wJSON.html#.parse has been implemented to facilitate writing and reading such data.

44See http://aide-line.inria.fr/build/www/client.html on the client side and on the server-side http://aide-line.inria.fr/build/www/httpService.html.

45The use is documented here: http://aide-line.inria.fr/build/www/Tabletop-driver.html and all hardware functions are available here: http://aide-line.inria.fr/build/www/Tabletop.html.

46The data structure and functions are documented here http://aide-line.inria.fr/build/www/Image.html.

and channels.

The main required image processing mechanisms are:

- Image rectification, i.e., perform a perspective transformation of the image to work on a gameboard top view, even if the camera is tilted.
- Image mapping, i.e., localize the present grabbed image for another reference image after the large translation of the setup wagon and small deformations (i.e., namely a 2D affine transform) due to optic imperfection, and light condition transformation.
- Thumbnail image recognition, i.e., recognizes among a predefined set of reference thumbnails corresponding to the different game card, which one (or none) has been set.
- Stable image detection, i.e., detect if something is moving (e.g., the player hands over the game board) or if the image is stable.

This has been implemented using standard image processing mechanisms since we are in a rather specific situation where known 2D objects have to be recognized. This only allows to partially estimate the 3D cubelets configuration in the fourth activity, and a companion project using machine learning mechanisms will complete the present implementation, beyond the present limits.

The key point here is that all functionalities are parameterized via a JSON data structure, so that they can be easily usable without imperative programming, e.g., using a web service.

Conclusion

This preliminary study allowed us to extensively experiment to which extent we can design and implement a low-cost tabletop set-up to collect learning analytics during computational thinking unplugged or tangible activities.

By low-cost, we mean the reproduction of the open hardware. For one activity, the order of magnitude is something like 100€ to 200€ of electronic components, depending on the activity, and 10€ of a laminated poster, the wood being easily obtained from recycled material. The software is free and open. The assembly time is about one day.

The software has indeed been designed to be efficient, reuse a maximal number of existing libraries, and rather simple to use, without any strong knowledge in C/C++ programming, just calling methods as in any other languages. Wrapping to Python or Javascript is proposed. It is fully modular to be reusable on other activity boxes.

After experimenting with several solutions, we concluded that the next generation will have a simpler hardware design, with the idea to use an external computer connected to the tabletop with fixed cameras, the processing being partially performed on the host computer. The software architecture is entirely distributed allowing this mutation with a minimal evolution of the code.
No mistake, this is only a prototype and this document only reports the first steps of design and construction of the setup and the related activities. Much work is still to be done to validate and experiment with the setup.

References

Alexandre, Frédéric, Jade Becker, Marie-Hélène Comte, Aurelie Lagarrigue, Romain Liblau, Margarida Romero, and Thierry Viéville. 2020. “Open Educational Resources and MOOC for Citizen

Understanding of Artificial Intelligence." https://hal.inria.fr/hal-03024034.
Banihashem, Seyyed Kazem, Khadijeh Aliabadi, Saeid Ardakani, Ali Delavar, and Mohammadreza Ahmadabadi. 2018. "Learning Analytics: A Systematic Literature Review" 9 (June). https://doi.org/10.5812/ijvlms.63024.
Cassone, Laura, Margarida Romero, Thierry Viéville, Cindy De Smet, and Mbemba Ndiaye. 2019. *Proceedings of the ANR #CreaMaker Workshop: Co-Creativity, Robotics and Maker Education.* https://hal.archives-ouvertes.fr/hal-02362121.
Clement, Benjamin, Didier Roy, Pierre-Yves Oudeyer, and Manuel Lopes. 2015. "Multi-Armed Bandits for Intelligent Tutoring Systems." *ArXiv:1310.3174 [Cs]*, June. http://arxiv.org/abs/1310.3174.
Curzon, Paul, Mark Dorling, Thomas Ng, Cynthia Selby, and John Woollard. 2014. "Developing Computational Thinking in the Classroom: A Framework." https://pdfs.semanticscholar.org/e4f3/c24924c5a707a196b2015494c829c15618d1.pdf.
Diamond, Adele, and Daphne S. Ling. 2016. "Conclusions about Interventions, Programs, and Approaches for Improving Executive Functions That Appear Justified and Those That, despite Much Hype, Do Not." *Developmental Cognitive Neuroscience*, Flux Congress 2014, 18 (April): 34–48. https://doi.org/10.1016/j.dcn.2015.11.005.
Gottlieb, Jacqueline, Pierre-Yves Oudeyer, Manuel Lopes, and Adrien Baranes. 2013. "Information-Seeking, Curiosity, and Attention: Computational and Neural Mechanisms." *Trends in Cognitive Sciences* 17 (11): 585–93. https://doi.org/10.1016/j.tics.2013.09.001.
Huang, Wendy, and Chee-Kit Looi. 2020. "A Critical Review of Literature on 'Unplugged' Pedagogies in K-12 Computer Science and Computational Thinking Education." *Computer Science Education* 0 (0): 1–29. https://doi.org/10.1080/08993408.2020.1789411.
Lachaux, Jean-Philippe. n.d. "Éduquer la métacognition, la clé du succès pour les enfants." cerveauetpsycho.fr. Pour la Science. Accessed November 15, 2020. https://www.cerveauetpsycho.fr/sr/ecole-des-cerveauxeduquer-la-metacognition-la-cle-du-succes-pour-les-enfants-15209.php.
Leeuwen, Anouschka van, Nikol Rummel, and Tamara van Gog. 2019. "What Information Should CSCL Teacher Dashboards Provide to Help Teachers Interpret CSCL Situations?" *International Journal of Computer-Supported Collaborative Learning* 14 (3): 261–89. https://doi.org/10.1007/s11412-019-09299-x.
Lodi, Michael. 2020. "Introducing Computational Thinking in K-12 Education: Historical, Epistemological, Pedagogical, Cognitive, and Affective Aspects." Theses, Dipartimento di Informatica - Scienza e Ingegneria, Alma Mater Studiorum - Università di Bologna. https://hal.inria.fr/tel-02981951.
Menon, Divya, Sowmya BP, Margarida Romero, and Thierry Viéville. 2019. "Going beyond Digital Literacy to Develop Computational Thinking in K-12 Education." In *Smart Pedagogy for Digital Learning*. Routledge.
Menon, Divya, and Margarida Romero. 2019. "Game Mechanics Supporting a Learning and Playful Experience in Educational Escape Games." In . https://doi.org/10.4018/978-1-7998-2015-4.ch007.
Menon, Divya, Margarida Romero, and Thierry Viéville. 2019a. "Going beyond Digital Literacy to Develop Computational Thinking in K-12 Education,." In *Smart Pedagogy of Digital Learning*, edited by Linda Daniela. Taylor&Francis (Routledge). https://hal.inria.fr/hal-02281037.
———. 2019b. "Computational Thinking Development and Assessment through Tabletop Escape Games." *International Journal of Serious Games* 6 (November). https://doi.org/10.17083/ijsg.v6i4.319.
Mukhopadhyay, S. C. 2015. "Wearable Sensors for Human Activity Monitoring: A Review." *IEEE Sensors Journal* 15 (3): 1321–30. https://doi.org/10.1109/JSEN.2014.2370945.
Nezami, Omid Mohamad, Mark Dras, Len Hamey, Deborah Richards, Stephen Wan, and Cecile Paris. 2019. "Automatic Recognition of Student Engagement Using Deep Learning and Facial Expression." *ArXiv:1808.02324 [Cs]*, July. http://arxiv.org/abs/1808.02324.
Prokofieva, Victoria, Svetlana Kostromina, Sofia Polevaia, and Fabien Fenouillet. 2019. "Understanding Emotion-Related Processes in Classroom Activities Through Functional Measurements." *Frontiers in Psychology* 10 (October). https://doi.org/10.3389/fpsyg.2019.02263.
Reimann, Peter. 2009. "Time Is Precious: Variable- and Event-Centred Approaches to Process Analysis in CSCL Research." *International Journal of Computer-Supported Collaborative Learning* 4 (3): 239–57. https://doi.org/10.1007/s11412-009-9070-z.
Romero, Margarida. 2017. "Les Compétences Pour Du XXe Siècle." *Usages Créatifs Du Numérique Pour l'apprentissage Au XXIe Siècle.. Québec: Presses de l'Université Du Québec*.
Romero, Margarida, Frédéric Alexandre, Thierry Viéville, and Gérard Giraudon. 2020. "Des

Neurosciences Computationnelles Aux Sciences de l'éducation Computationnelles Pour La Modélisation Du Cerveau de l'apprenant et Du Contexte de l'activité d'apprentissage." *Bulletin de l'Association Française d'Intelligence Artificielle*, no. 108: 24–27.

Romero, Margarida, Dayle David, and Benjamin Lille. 2018. "CreaCube, a Playful Activity with Modular Robotics." In *Games and Learning Alliance*. Palermo, IT.

Romero, Margarida, and Marie Duflot. 2018. "À Gauche Ou à Droite Du Robot ? Test de Perspective Décentrée Gauchedroite Par Le Biais d'une Activité Sur Papier et d'une Activité de Robotique Pédagogique." In *CIRTA 2016*.

Romero, Margarida, Marie Duflot-Kremer, and Thierry Vieville. 2019. "Le Jeu Du Robot : Analyse d'une Activité d'informatique Débranchée Sous La Perspective de La Cognition Incarnée." *Review of Science, Mathematics and ICT Education* 13 (1): 35–49. https://doi.org/10.26220/rev.3089.

Romero, Margarida, Alexandre Lepage, and Benjamin Lille. 2017. "Computational Thinking Development through Creative Programming in Higher Education." *International Journal of Educational Technology in Higher Education* 14 (December). https://doi.org/10.1186/s41239-017-0080-z.

Romero, Margarida, Benjamin Lille, Thierry Viéville, Marie Duflot-Kremer, Cindy de Smet, and David Belhassein. 2018. "Analyse comparative d'une activité d'apprentissage de la programmation en mode branché et débranché." In . https://hal.inria.fr/hal-01861732.

Romero, Margarida, Stephanie Noirpoudre, and Thierry Viéville. 2018. "Que Disent Les Sciences de l'éducation à Propos de l'apprentissage Du Code?" *Revue de l'association EPI*, no. 2015 (May). https://www.epi.asso.fr/revue/articles/a1805b.htm.

Wing, Jeannette M. 2011. "Computational Thinking." In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 33. IEEE.

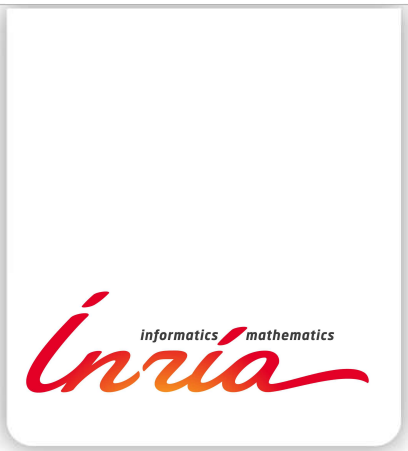
Contributions and acknowledgements

- *Sabrina Barnabé* designed the tabletop mechanism, constructed the tabletop hardware and participated in the system conception, calibration and validation.
- *Lola Denet* developed the final French version of the scenario, designed and realized the user-interface software of the storyline scenario.
- *Mathieu Manrique* created and realized the graphics of the user-interface software and co-developed the French version of the scenario.
- *Divya Menon* is the instructional designer for this project who, at the origin of this project, provided the pedagogic and didactic aspects of the scenario, and performed a state-of-the-art review related to addressed issues. She has also done an editorial review of this paper.
- *Éric Pascua*l designed the electronic hardware, realized the hardware drivers and advised the set-up realization all along with the project.
- *Margarida Romero* is the educational science researcher at the origin of this project, providing the theoretical ideas and framework, the pedagogical elements, and advising the didactic incomes, and supervising Divya Menon work.
- *Thierry Viéville* is the informatics researcher at the origin of this project, contributed to the didactic aspects of the setup in link with science outreach issues, realized the algorithmic software and took care of the project coordination.

The present paper input has been provided by all co-authors, shaped by the two last authors, and revised by the whole team.

The present realization has been funded by the Inria Science Outreach Project (*Mission de Médiation Scientifique Inria*) https://pixees.fr/mediation-scientifique-inria and the *Class´Code project* https://classcode.fr, within the scope of their support to research development in link with their science popularization actions.

The present realization has been supported by the LINE laboratory and the PoBot maker space and is now part of the Inria Mnemosyne AEx AIDE action https://team.inria.fr/mnemosyne/aide and the ARN CreaMaker project https://creamaker.wordpress.com.



RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399