So, H. J. et al. (Eds.) (2020). Proceedings of the 28th International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

Playful Learning and shared Computational Thinking: the PaCoMa case study.

Andrea VALENTE[a*], Emanuela MARCHETTI[b]
[a]*Game Development and Learning Technology, University of Southern Denmark, Denmark*
[b]*Dept. for the Study of Culture, Media, University of Southern Denmark, Denmark*
*anva@mmmi.sdu.dk

Abstract: We know from previous studies that tangible kits support lively and playful forms of shared sense-making better than digital, computer-based solutions. A typical problem that we observed, when groups of learners work on a shared activity at a computer, is a tendency to work one at the time, often leaving less expert or less active learners at the margins. However, we observed that tangible materials such as paper or fabrics, afford more naturally face-to-face engagement in small groups, allowing for eye-contact, dialogue and active engagement. In this paper we focus on playful learning activities about Computational Thinking, for early primary school pupils, and aim to better define the requirements for a new kind of tangible learning materials that we call PaCoMa, short for Paper Computing Machines. Our research questions deal with the role of tangible play in CT learning, and tangible play in shared sense-making. This paper describes our theoretical and design-based explorations of how to design an activity popup book to introduce kids 6 to 11 years old to CT. At a practical level, we discuss how paper-based mechanisms can represent the state of a variable or a machine, how to generate random symbols, how to embed code-like rules about jumping from state to state, as well as how to represent sequences of instructions/choices as physical objects. Armed with these mechanisms, we discuss the design of exemplar paper machines and propose scenarios of use in shared CT learning activities. Our scenarios address the emergence of shared affinity spaces, which we consider a precondition for simulative play, and to provide room for negotiation and creation of meaning through the paper machines. To assess the potential benefits of this tangible, analogue approach, qualitative tests are being organized with local and international institutions.

Keywords: Computational Thinking, primary school, playful learning, tangibles, analogue computation

1. Introduction

In the past 15 years we have been working on what is now called Computational Thinking (CT for short), and we have gained significant experience in collaborating and supporting playful learning activities in primary schools and non-scholastic institutions in Denmark (Valente, 2004, Marchetti & Valente, 2015, Valente & Marchetti, 2017, Valente & Marchetti, 2019, and Pedersen et al., 2020). Our interest focuses on the intersection between playful learning (PL for short) and CT, and we investigated both technological solutions (such as Valente & Marchetti, 2019, and Pedersen et al., 2020) and analogue, tangible kits and approaches for playful and active learning, like in Valente & Marchetti (2017).

During our previous studies, we noticed that tangible kits seem to support more lively and playful forms of shared sense-making than digital solutions, especially those used on a regular computer. When working on a computer on shared face-to-face activities, learners tend to work one at a time: often one codes while the others watch. As a result, the most proficient learners are also the most active, while those in need of more practice remain less active, hence not overcoming their difficulties. On the other hand, we observed that tangible materials (such as paper or fabrics) afford more naturally face-to-face engagement in small groups, allowing for eye-contact, dialogue and active engagement (Pedersen et al., 2020).

In this paper, we want to take a closer look at play in the context of playful learning activities aimed at CT, to better define the requirements for a new kind of tangible learning materials that we call PaCoMa, short for Paper Computing Machines. In this respect, our research questions deal with:

- The role of tangible play in CT learning,
- Tangible play in shared sense-making

This paper describes our theoretical and design-based explorations of the affordances of how to design an **activity popup book** to introduce kids 6 to 11 years old to CT: the idea is to address these pupils, which are in the concrete operational stage according to Piaget (as also discussed in Bers, 2020), and engage them with a series of DIY, hands-on tinkering exercises, using popup tangibles to play and learn about CT, alone or in small groups.

The rest of the paper presents our reflections and findings about playfulness, game-based learning and the challenges of adopting them within CT (section 2). The design process behind the PaCoMa activity popup book is presented in section 3, together with a discussion of scenarios of use. Section 5 concludes the paper and presents ongoing and future work.

2. Playful learning for CT

PL is defined as an experience harmoniously combining play and learning, in which players acquire new knowledge through play. Play is itself a free, ambiguous experience, centered on the players' goals and defying clear definitions. However, play is defined as a transformative practice, through which players create their own world through their actions, transcending their present circumstances as if they were hallucinating (Gee, 2017, Vygotsky, 1978). In PL players act as if they were simulating themselves the situation depicted in the game, imagining themselves inside the game, leading them to reproduce specific aspects from the targeted knowledge domain (Gee, 2017). Hence, the players engage in problem solving, requiring application and in-depth reflections of the targeted knowledge (Vygotsky, 1978, Gee, 2017).

In order to foster such experience in PL, play shifts from a free to a regulated practice, framed within the constrained systems of rules of the adopted game (digital or analogue), yet still aiming at eliciting fun and a feeling of gratification (Bogost, 2016). Differently from free play, games are centered around a challenge, through which players experience "*game breakdowns"* (Benton et al., 2019), conceptual or systemic contradictions to be solved. Breakdowns can be generated by players' unsuccessful actions in the game, due to lack of understanding on what to do next or lack of involvement, due to boredom or frustration (Benton et al., 2019). Facing such breakdowns, players are forced to develop strategies to achieve "*game breakthroughs"* (Benton et al., 2019)*,* which are players' achievements in problem solving, through the use of acquired knowledge.

In this way, as players learn how to play a game, they are learning the knowledge and skills targeted by the PL activity they are attending. Players are facilitated in their learning process by instructional design, narrative, rules, and mechanics embodied in the game challenges. Rules and mechanics provide a constraining framework, forcing the players to acquire the targeted knowledge and skills to win the game, while hindering them from finding shortcuts. In this sense, game rules and mechanics act as resources for pedagogical alignment (Biggs and Tang, 2011), as by winning the game the players are supposed to fulfill the goals of the game as well as the learning goals.

Interestingly typical PL applications for CT, such as MIT Scratch, Google Blockly or BBC Micro:bit, are aimed at a single player/learner, more or less explicitly. This might also be a consequence of the fact that such applications are developed to be used on computers and laptops, which do not easily enable more than one individual at a time actively using the system. Through previous studies, we noticed that when a group of learners is given a task to complete on the computer, only one learner actively engages with the system, while the others sit around and participate in a more passive way. Typically the active learner is also the one who is more proficient than the others in solving the given task, acquiring a leading role, hence the one who has less need to learn is the most active, while those who would need more practice, end up acting in more passive roles. On the other hand, PL is supposed to benefit from the social dimension of play, acknowledging that social play contributes to the emergence of shared forms of meaning making. Through social play, players inspire each other, engage in competition or collaboration, creating *affinity spaces* (Gee, 2017), imaginary worlds delimited by the participants' presence, actions, and physical or virtual context of interaction. We find that in PL these affinity spaces provide room for creation and negotiation of meaning, in which the players can inspire each other in embracing new ideas or possibilities for winning their game, also supporting each other when facing game breakdowns or simply issues when lacking understanding of the game or of the digital system available. We have previously found (Pedersen et al., 2020) for instance that design thinking activities and interaction with tangible, analogue materials like fabrics to make interactive puppets, better support social sense-making than simply working individually on the computer. Based on such insights, we aim at exploring forms of analogue CT, supported by shared forms of tangible interaction and design thinking.

In fact, one of the goals of CT is to introduce learners to a computational way of thinking and solving problems (Wing, 2006). Interestingly, when we consider professional programmers (e.g. game developers) we notice that their development methodologies usually involve plenty of low-fidelity prototyping, typically with paper or simple materials like LEGO bricks or modeling clay. In this paper, we advocate to leverage on such techniques also for pupils, which in our experience with Danish educational institutions are already used to tinkering as part of their learning. Since we want to scale down CT and turn it into a playful activity, suitable for young pupils, we looked at Paper's constructivism and attempted at demystifying computation by showing how to remake computing machines as tangible, manually operated devices. This scaling down also requires avoiding or at least seriously reducing the dependency of CT on math; in this paper we propose to soften the math requirements of CT by looking at computation as manipulation of symbols, instead of a way to perform calculations.

We have conducted a preliminary survey of activity books for 6 to 10 years old, and we found many examples, such as McManus & Magar (2017), a successful book translated in multiple languages in Europe; however, these materials are intended for play and work at home, typically not adopted by institutions, and have very simplistic and superficial CT contents. Interestingly many of the books we surveyed suggest continuing the paper-based activities as Scratch projects.

Finally, we see our idea of designing paper computing machines as a way to implement tangible and playful *notional machines*. A notional machine is *"[...] an abstraction designed to provide a model to aid in understanding of a particular language construct or program execution. The notional machine [...] presents a higher conceptual level by providing a metaphorical layer above the real machine [...] that are hoped to be easier to comprehend than the real machine."* (Berry & Kölling, 2014). Interestingly, a notional machine can be defined in many different ways, from a visual representation implemented as software, to a simple set of rules and tangibles operated by learners (like the tabletop games about CT analyzed in Scirea & Valente, 2020). Looking at board games to support playful learning of complex phenomena and systems is not a new idea (see for instance in the inspiring approach in Lin & Hou, 2016); however, here we propose to adopt a similar approach for CT and for young pupils, which presents specific problems, as discussed in the next section.

3. Design of PaCoMa's activity popup book

The PaCoMa activity book should work similarly to *print-and-play* games (Ferdinands, 2019), that can be printed, eventually customized, cut and folded, to become tabletop games for one or more players. In the case of PaCoMa, the pupils and their teachers will print and play, i.e. they will build computing machines with paper, then play with them in groups, to solve a possible computational problem, and reflect on how computation works. As they get more proficient with these machines, the learners should also be able to modify them, customize them, to solve other, related problems; and finally they will be able to design new computing machines to work with new, possibly more complex problems. To achieve this, we need to look at which topics within CT we can cover with the activities in the book, then which paper mechanisms can we propose to our target group, for their tangible play and paper-and-scissors activities.

We have been working on a structure for the PaCoMa activity book that focuses on playing with computation and communication, abstracting from programming languages and hardware considerations. The topics we propose to cover, organized to make sense for learners, are:

1. Simple state machines. Example of activity: a lock that opens using a password;
2. Turing Machines. Example of activity: a machine that counts in unary or that draws patterns of colored symbols;
3. Create and program robot-pets. Example of activity: define and enact the behavior of a paper robot that moves and collects items;
4. Communication of images, introduction to Computer Graphics. Activity: send and receive an image by going from pixels to numbers and back;
5. Machines that generate random symbols. Activity: machine-generated music;
6. Programming of non-linear games. Activity: design, create and play a simple video game;
7. Scrambled communication. Activity: send messages with probability of errors, find ways to fix them at reception.

The actual conception order of these topics was: topic 2, i.e. Turing Machines, and topic 3, Robot-pets, then by way of simplification, state machines. Since we wanted some form of programming in our activities, but not necessarily coding or block-coding, we added topic 6, non-linear games, as a more software-oriented version of topics 2 and 3. Finally, to cover more than just visuals, we added topic 5, music generation. Topics 4, 5 and 7 are meant to show the pupils that multimedia (here images, audio and messages in general) are really just a matter of encoding and decoding of symbols, possibly with transmission channel in between.

We are aware that these seven topics might look very theory-heavy and are usually not an explicit part of CT, also considering that our target group is young pupils in primary school. However, according to some definitions (e.g. Tedre and Denning, 2016) CT learners should be presented with the basic concepts of the imperative programming paradigm: variables, conditionals, loops and functions. And it is well known in theoretical Computer Science that the imperative paradigm originated from Turing Machines. Furthermore, the simple kinds of automata we are considering in our activity book are based on test-and-jump rules, i.e. they require reasoning in the same way as with conditionals in CT, but with a more *mechanical* flavor than in programming. Moreover, we have been pursuing a surprising relation between theoretical Computer Science and tangible games for a long time (Valente, 2004), and here we maintain that many theoretical models of computation can indeed be turned into games, with rules and mechanisms, and manually emulated to better understand them; this is supported by the concept of notional machines, since in fact tokens and rules of a board game can represent the state and logic of a computational device, and act as a notional machine for such device.

Apart from imperative programming, CT also requires familiarizing learners with algorithms, foster problems solving skills, and working with data. The proposed topics in our activity book support this because they require iterative and algorithmic thinking, through design and enactment of the behavior of paper machines.

3.1 Exploring paper mechanisms

To be able to implement the topics in our activity book, we need paper-based mechanisms to: remember and change the state of a variable or a machine, generate random symbols, be able to jump from state to state, and to represent sequences of instructions/choices as physical objects. We explored a few possible ways to represent the state of a variable: a board with a marker, foldable tabs, a deck of cards, and a slider.

A board with a marker (see Figure 1.1) is a sheet of paper showing all possible values of the variable, and a marker-object to keep track of the current state. As an example, consider a Bingo card. This solution works only if a flat, horizontal surface is available, and requires markers as separate objects from the value sheet, which could easily be lost or mistakenly moved, resulting in loss of information about the state.

Foldable tabs (as visible in Figure 1.2). A sheet of paper with a linear sequence of possible values; one side of the sheet (possibly the top) has tabs that can be folded towards the values, to indicate which one is the current state. This solution works even if the sheet is a page of a book and has no separate parts. However, the user has to remember that only one tab can be folded at any given time, so there is more possibility of human error than with the previous solution. A variation of solution 2 can be used when only 2 values are possible (see Figure 1.2B), and we call it foldable binary value.

A deck of cards solution is reminiscent of a physical display: a deck is printed with a card per possible value of the state. The user will simply stack the cards, being careful to maintain the top card on the deck as the current state of the variable (see Figure 1.3). This solution has the disadvantage of being less automatic, requiring some planning from the part of the user; moreover, the cards might require a horizontal surface to be usable and this solution, like the first, has parts that can be lost.

And finally, a slider: a popup paper device with a cursor-like part moving along a list of values (see Figure 1.4). The user slides the cursor up and down (or sideways if the orientation of the slider is horizontal). This solution is quite stable, automatic and has no separate parts, because the sheet of paper with the list of values cannot be removed from the cursor. A slider can be embedded in a book page if needed.

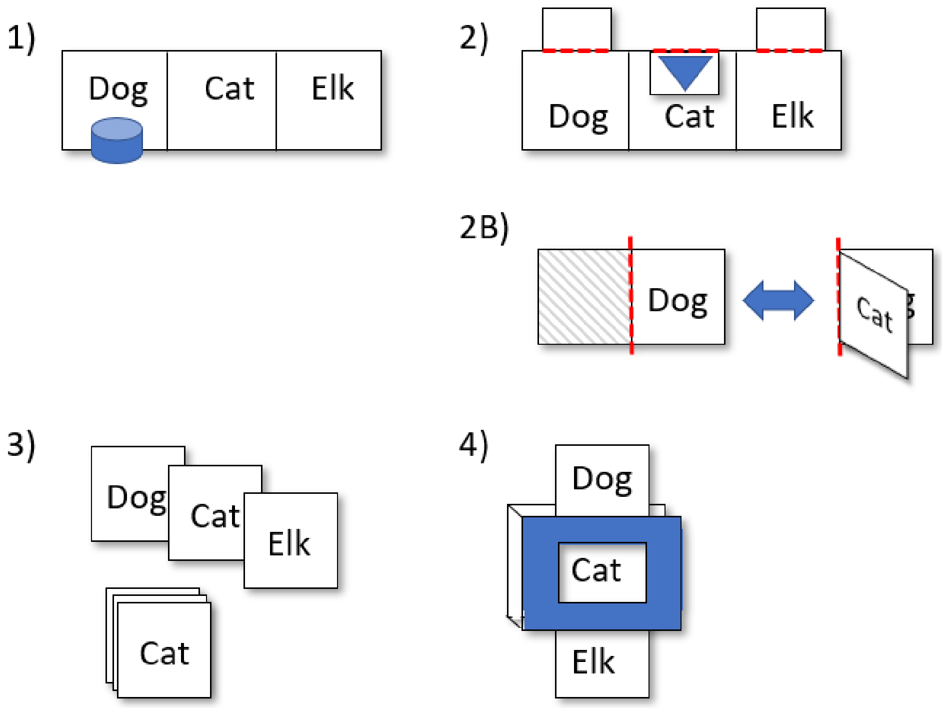

Figure 1. Alternative designs for tangible representation of state.

All of these solutions can be combined to describe more complex states. For example, two decks of cards can be stacked side by side, the first with cards from 0 to 4 and the second with cards from 0 to 9. These cards can be used to represent any number from 00 to 49, and the two cards in the two decks could be color-coded to avoid mixing them. Alternatively, two foldable tab sheets could be used, one with the digits from 0 to 4 as values, and the second with digits from 0 to 9.

For states that have many component values (called *records* or *objects* in Computer Science) a combination of various solutions could be possible: e.g. in an role-play game a character might be described by a foldable tab sheet with values "Warrior", "Mage" and "Rogue", followed by a slider showing the energy, with perhaps 5 values from "low" to "high", and a deck of cards representing which weapon, among a few available, is the character currently holding. The result would be a tangible dashboard, representing the complex state of the character at any moment during a game. Scirea & Valente (2020) discusses similar composite solutions found in board games.

Random generation is needed typically to simulate some level of Artificial Intelligence. The board game community offers many examples of tangible randomization solutions (Scirea & Valente, 2020), usually connected to the need of defining behavior of interesting opponents, or to surprise the player by generating parts of the game on the fly, in unforeseeable ways (in computer games this approach is called Procedural Generation, and can be used with different contents, such as maps, images, characters, and even music). Here we are interested in methods to generate random values that can be used by primary school pupils, so they have to be very simple to deploy and use, but they can of course be based on probability or statistics; we just have to hide the complexity and the math and embed them in learner-friendly paper implementations. In our explorations we have recognized the following random generation mechanisms: fishing a card from a deck, chain-fishing, and pre-randomized tables.

Fishing a card from a deck is a simple mechanism that works like a die: the user has a deck of cards with one card per possible value, the deck gets shuffled, and the user fishes one card. The advantage of cards over dice is that cards can easily be added to the deck, altering the probability of fishing the possible values. For example, using a deck with cards “Dog”, “Dog” and “Cat”, the user has twice as many probabilities of fishing a "Dog" card than a "Cat" card. The main limitation of this approach is that the random values generated are independent of each other, i.e. the random generator has no context awareness.

To solve the problem of context of the previous solution, we can organize the cards in many decks, and let the first card fished decide which following deck will be used to fish the second card, and so on, i.e. chain-fishing. Consider the simple setup where we want to randomly generate all possible combinations of two decks: one, red deck, with cards "Dog" and "Cat", the other blue deck with cards "S", "M", "L" (i.e. the size of the animal). The user can be given two decks and fishing one card from each will provide any random pair animal-size. However, we might want to forbid certain combinations: perhaps for us dogs can be of any size, but cats can only be "S" or "M". To achieve this, we can duplicate the blue deck, obtaining the blue1 and blue2 decks, and remove some certain cards from each new deck. The blue1 deck will only have "S", "M", and "L", while the blue2 deck will have "S" and "M" only. And now we just have to associate each card in the red deck with one of the two new decks. The result will be three decks: the red deck with "Dog-blue1" and "Cat-blue2", the blue1 deck with all 3 sizes, and the blue2 deck with only small and medium sizes. The user will have to remember to always start by fishing a card from the red deck, then following the instruction of that card, fish a card from either the blue1 or blue2 deck. And this mechanism can be easily scaled to more values and longer sequences. Note that this solution is informed by Markov Chains, and allows the designer to create rather smart random generators that are still easy to use and do not require any math to operate. However, the user has to be careful to use the decks in the correct sequence and not to lose or mix cards; and it could be difficult to use this mechanism without a horizontal surface.

A pre-randomized table is a printed grid with certain values, eventually repeated and randomly positioned in the grid; without looking, the user points at the table with a finger or a pencil, selecting a random value. The main disadvantage of this mechanism is that there is no tangible representation of the generated values, so if the user has to generate and remember multiple values, she will have to keep notes using separate mechanisms. This solution is very easy to embed in books and can be used in a modular way by providing two or more pre-randomized tables side-by-side, and ask the user to use them in sequence. It is important to notice that even if the examples above use names or numbers, these mechanisms work perfectly also when a value is an icon or any symbol; so when designing a random generator for maps, the individual cards can simply be images of parts of a map (perhaps a tile with a river, one with houses, etc.). Finally, to be able to build paper computing machines we need ways to represent jumps in the flow of execution (called *conditionals* or *IF-statements* in CS) and sequences of instructions. Sequences of instructions are perhaps simpler to implement, since they can be realized by using the slider solution discussed in this section. Instead, jumps can be more complicated; in board games like the goose game, the board itself is divided into areas or cells and simple instructions are written in some of the cells. Those instructions can be to jump to a different cell or to perform certain actions (e.g. roll a die) to continue the game. From a Computer Science point of view, the board acts as a graph and graphs are indeed a classical way to represent the behavior of a computing machine (or in the case of the goose game a set of rules for a board game). However, graphs are a sophisticated notation that is usually introduced only at university level and requires specialized forms of algebra to correctly reason about. For instance, the state machines we want to present in our activity book are classically visualized as graphs in CS books. One of the main problems of working with tangible realization of graphs, is that graphs might not be *planar*, meaning that there can be arrows (i.e. edges) connecting nodes (the states of the machine that the graph represents) in circular and intersecting ways, making it impossible to draw the graph on paper without any arrows crossing. Moreover, in a graph the physical distance among nodes is irrelevant, a property which is very difficult to represent with tangibles.

An alternative way to show a graph is to use a table. For example, the rules of a board game are typically expressed as a sequence of statements, possibly with conditions and consequent actions; those sequences can be mapped into a table with condition/action pairs, such as the following (or a graph in general):

IF condition_i1 THEN Action_j1 | IF condition_i2 THEN Action_j2 | etc.

A table like this can be written and given to the users of our paper computing machines, but in certain cases we might need the table to be changeable by the users. In this case we could implement a tangible version of the table by using two decks of cards (or two instances of our foldable tabs, like we discuss above). One deck will contain all condition cards, while the other deck has all possible actions; the user will just fish one card from each deck to create one row of her table. Interestingly, this mechanism can be used to both setup a fixed table of rules, but using reusable tangible elements (i.e. the cards) or to randomly generate strange, surprising tables that will result in strangely behaving computing machines.

3.2 Paper machines, scenarios of use, and discussion

Now that we have defined a few alternative ways to implement the state of computing machines, randomization algorithms, jumps and sequences instructions, in tangible form, we will present a few examples of designs for such machines: the Popup Turing Machine (PopTM in the following) and the Mintempo tangible game console (on the left and right respectively of Figure 2). A Turing Machine (TM for short) is a famous theoretical computing machine (invented by Alan Turing in his seminal 1936 paper), often considered the original conceptual model for all modern computers. A TM computes by using a tape where input symbols are written by the user, then it follows a list of simple instructions to read and re-write the symbols on a tape, producing a new series of symbols that constitute the output.

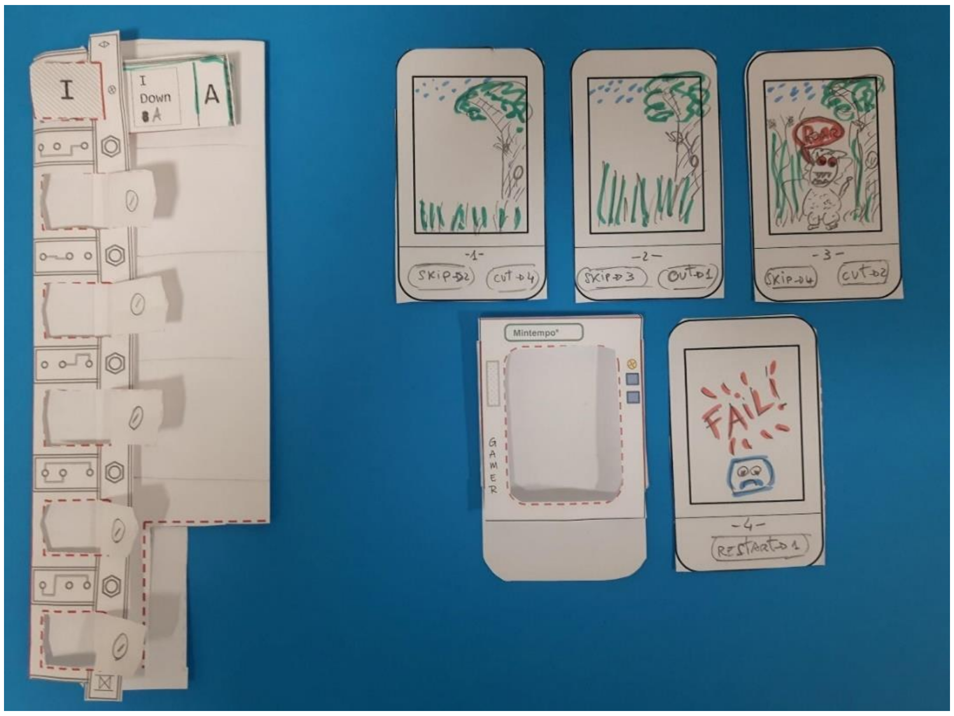

Figure 2. Two paper computing machines: a paper Turning machine (left) and a tangible portable console that we call Mintempo (center and right).

Our PopTM works in the same way: Figure 2 (on the left) shows the tape the machine, oriented vertically, with the input I00000 (the symbols as they appear from top to bottom). The program of this particular machine is very simple:

- Start from instruction A
- (A) if the current tape symbol is "I", then *rewrite* it as "I", *go down* to the next symbol, and stay, i.e. *jump* to instruction A; otherwise, *rewrite* the current symbol as "I", go *down* to the next symbol, but *jump* to instruction B
- (B) if current symbol is "I", then *halt*; otherwise, *halt* as well

The user will have to place the piece of paper representing the program under the first symbol on the top of the tape, with the program set to instruction A. Because the tape reads I00000, the computation will change it, step by step, until the machine halts and the tape shows II0000; so the machine started with one "I" symbol in input and halted with two "I" in output, hence, the machine added one to a number expressed in *unary notation*. It is easy to write new programs for the PopTM and calculate other mathematical or logical operations, such as additions or subtractions.

Even if the typical examples of use of a TM are mathematical, we have also designed examples in which the PopTM is used to change symbols in a more aesthetic way, or to work on letters and words. Our earlier experiments with CT sense-making and TMs are presented in Valente & Marchetti (2011). Here we propose an analogue and playful reinterpretation of TM, which aims at supporting shared sense-making in small groups of players through simulative forms of play (Gee, 2017, Vygotsky, 1978) encouraging them to enact the computational mechanisms of TMs, as if they were *inside* a TM, but without dealing with the algebraic formalism. In Valente & Marchetti (2011) we observed primary school pupils making sense together of the symbols and sequences of actions to be performed by the paper machine, hence supporting each other when encountering difficulties (Benton et al., 2019). Moreover, having explicit computational rules on paper, forced the players to articulate step by step the correct steps, eliciting in-depth reflections, instead of taking the system for granted as they would do with a digital game.

Figure 2 (on the right) shows a tangible, manually operated, game console that we call "Mintempo", which in Danish sounds like "my own pace". This paper machine is inspired by famous portable digital consoles such as the Nintendo Gameboy or DS, but instead of storing a game on a cartridge, we decided to use a deck of cards per game. The machine in Figure 2 has a game made of a deck with four cards; each card is numbered (from 1 to 4) and the lower part of each card has button-like jump instructions, which represent the program of the game. To play the game a player should start from card 1, and decide if she wants to follow the "skip->2" or "cut->4" instructions. Taking the "skip->2" option, the player will change the current card with card number 2, which shows the jungle grass growing, and provides other possible options. Choosing "cut->4" instead will send the player to card 4, which ends the game with a "fail" message, since there was no grass to cut at this point in the game. Looking at the cards from a CT point of view it is clear that they represent the program of the game and that they form a graph, with nodes (i.e. the cards) and edges (i.e. the jump instructions). Writing a game for the Mintempo console means then, to draw cards with all possible states of the game, and connect them together in a graph, by using jumps. The game mechanics emerge when a player executes the jumps in her preferred way. With Mintempo the players are encouraged to simulate on paper the experience of playing a digital game on an actual portable console, and also reason about embedded coding for that console. Our paper console leverages on game mechanics from card games, to suggest possible interaction and redesign of the system, combining play with design thinking. As for the PopTM, also in this case the goal is to lead players to simulate with each other (Gee, 2017, Vygotsky, 1978) gaming and coding experience afforded by digital systems. With both the PopTM and the Mintempo paper console, we aim at forcing players to articulate in detail the steps required to play a game, or reason about game design so that other players will take the appropriate steps and be able to play new games.

We propose to use these machines in class as support materials for hands-on tasks aimed at small groups of pupils engaged in shared sense-making. In this scenario, the teacher can prepare a few PopTM or give a working game (in the form of a deck of Mintempo cards) to a class of pupils, divided into small groups; the groups should be introduced to the machines using a use-modify-create approach, where a working paper machine is constructed (from printed templates) together with the pupils, and then demonstrated. The pupils can then be asked to alter the machines, for instance by customizing them, by coloring or re-drawing them. Finally, the groups could be given tasks that need to create new machines, or new programs for their machines, and present/challenge other groups or the teacher(s). These are emergent forms of shared sense-making that we experienced before in our studies with Danish primary school classes, and they are usually associated with tangibles, tinkering and playful learning scenarios.

In more advanced classes the same type of scenario can be extended to include exercises, where the groups are asked to implement a digital version of their tangible creations, by using MIT's Scratch or similarly suitable programming environments. A Mintempo game, for example, can be very easily digitized, by scanning the cards' images into Scratch, and then by writing simple code that implements the jumps among cards, following the instructions in the tangible cards. We consider this as a way to bridge analogue with digital CT, as well as shared with individual CT, since the resulting digital games would have been developed as shared artefacts but played as single-player Scratch games.

The paper-based computational machines we designed for our PaCoMa activity book aim at providing support for affinity spaces (Gee, 2017), to enable the players to create their own world, in which simulate and reflect through social play and design thinking on computational concepts, which preexist the definition of current software and hardware. We see the emergence of shared affinity spaces as a precondition to support simulative play as well as providing room for negotiation and creation of meaning through the paper machines. Moreover, we aim at observing if through our paper machines, the players might achieve a *meta-level awareness* of CT's fundamental concepts (such as variables, conditionals, loops, etc.): we would like to convey to our players/learners the notion that CT goes beyond a specific software or hardware system, and that the design of computational systems offers a lens to look at and understand problem solving in the modern world.

4. Conclusion and on-going work

Looking back at our experience with CT-related projects with schools and afternoon clubs, we believe that a DIY approach with cheap tinkering materials should work very well for early primary school pupils engaging in PL activities, targeting CT and support their teachers.

In line with constructivism, we believe that providing kits to build a computing machine out of paper, will foster feelings of empowerment and motivation, and provide greater understanding of CT topics. Our design of PaCoMa activity book can be seen as a prototype for a new kind of tangible materials, which have the potential to scaffold shared sense-making of CT, its principles and practices. We feel that demystifying computing machines and algorithms is perhaps the most valuable effect that CT can have on the new generation.

We are currently organizing tests of PaCoMa with our local and international network, and we expect to start in the coming fall semester. The tests will be conducted as qualitative studies (similar to what was done in Hsieh, Yi-Chun & Hou, 2015), adopting co-design with older learners in afternoon clubs and at the same time observing playing sessions with groups of younger pupils. The investigations will shed light on practical aspects of orchestration of classroom-wide play sessions with PaCoMa, to better support teachers, and help us better understand the interaction between tinkering with paper computing machines and existing school subjects.

References

Benton, L., Vassalou, A., Barendregt, W., Bunting, L, & Revesz, A. (2019). What's Missing: The Role of Instructional Design in Children's Games-Based Learning.CHI 2019, May 4–9, 2019, Glasgow, Scotland UK, Paper 582

Berry, M., & Kölling, M. (2014). The state of play: a notional machine for learning programming. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 21-26).

Bers, M. U. (2020). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge.

Biggs, J and Tang, C. (2011). Teaching for Quality Learning at University, McGraw-Hill and Open University Press, Maidenhead

Bogost, I. (2016). Play anything: The pleasure of limits, the uses of boredom, and the secret of games. Basic Books.

Ferdinands, B. (2019). 49 Free Print & Play Games. In the online magazine Boooored, last visited 26th August 2020, url: https://boooored.com/printandplay/49-free-print-play-games/

Gee, J.P. (2017). Affinity Spaces and 21st Century Learning. *Educational Technology*, March-April 2017, Vol. 57, No. 2 (March-April 2017), pp. 27-31

Hsieh, Y. H., Yi-Chun, L., & Hou, H. T. (2015). Exploring elementary-school students' engagement patterns in a game-based learning environment. Journal of Educational Technology & Society, 18(2), 336.

Lin, Y., & Hou, H. (2016). The Design of an Ecosystem-Education Board Game Integrating Role-Play and Peer-Learning Mechanism and Its Evaluation of Learning Effectiveness and Flow.

Marchetti, E., & Valente, A. (2015). Learning via Game Design: From Digital to Card Games and Back Again. *Electronic Journal of E-Learning*, *13*(3), 167-180. http://www.ejel.org/volume13/issue3/p167

McManus, S., Magar, R. (2017) *Coder Academy: Are you ready for the challenge? Ivy Kids publisher, ISBN1782405038.*

Pedersen, B. K. M. K., Marchetti, E., Valente, A., & Nielsen, J. (2020). Fabric Robotics - Lessons Learned Introducing Soft Robotics in a Computational Thinking Course for Children. In *Lecture Notes in Computer Science (LNCS)* Springer. https://doi.org/10.1007/978-3-030-50506-6_34

Picard, R.W., Papert, S., Bender, W., Blumberg, B., Breazeal, C., Cavallo, D., Machover, T., Resnick, M., Roy, D., & Strohecker, C. (2004). Affective learning—a manifesto. BT technology journal, 22(4), 253-269.

Scirea, M., Valente, A. (2020). Boardgames and Computational Thinking: how to identify games with potential to support CT in the classroom. In *Proceedings of Foundations of Digital Games (FDG2020)*

Valente, A. (2004). Exploring Theoretical Computer Science Using Paper Toys (for kids). In *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT'04)* (s. 301-305). IEEE Computer Society Press. https://doi.org/10.1109/ICALT.2004.1357424

Valente, A., & Marchetti, E. (2011). Programming Turing Machines as a game for technology sense-making. In *Proceedings of 11th International Conference on Advanced Learning Technologies (ICALT), 2011 IEEE.* (pp. 428-430). IEEE Computer Society Press. https://doi.org/10.1109/ICALT.2011.134

Valente, A., & Marchetti, E. (2017). From Cards To Digital Games: Closing the loop. In *Proceedings of the 6th International Congress on Advanced Applied Informatics LTLE 2017* (s. 507-510). IEEE. https://doi.org/10.1109/IIAI-AAI.2017.50

Valente, A., & Marchetti, E. (2019). The Road towards Friendly, Classroom-Centered Interactive Digital Contents Authoring. I M. Chang, H-J. So, L-H. Wong, J-L. Shih, & F-Y. Yu (red.), *Proceedings of the 27th International Conference on Computers in Education: Taiwan: Asia-Pacific Society for Computers in Education* (Bind 2, s. 38-46). Asia-Pacific Society for Computers in Education. http://ilt.nutn.edu.tw/icce2019/dw/ICCE2019%20Proceedings%20Volume%20II.pdf

Vygotsky, L. S. (1978). Mind in Society. Harvard University Press

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.