Explorations in Computational Tinkering

by
Carmelo Presicce

Bachelor of Science in Electronics Engineering, Università di Bologna, 2004
Master of Science in Computer Engineering, Università di Bologna, 2008

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the
Massachusetts Institute of Technology

September 2017

© Massachusetts Institute of Technology 2017. All rights reserved.
Explorations in Computational Tinkering

by
Carmelo Presicce

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

In recent years, there has been a growing interest in tools and strategies to support computational thinking, emphasizing a systematic application of computational ideas to problem solving. In this thesis, I focus instead on computational tinkering, exploring tools and strategies to support playful and iterative experimentation with physical and computational materials. I propose two different ways to look at computational tinkering: bringing the playful attitude of tinkering to building with code, and integrating computation into physical tinkering activities. Through four case studies, I describe the design and facilitation of activities for children in different contexts, using a particular tool - LEGO Programmable Art Machines. For each case, I share my observations and reflections about children engaging with the activity, to iteratively improve the design of the experience. I conclude by identifying some key characteristics of learning through computational tinkering and proposing directions for further work in the design and outreach of computational tinkering activities and ideas.

Thesis Advisor

Mitchel Resnick
LEGO Papert Professor of Learning Research
Explorations in Computational Tinkering

by

Carmelo Presicce

Signature redacted

Karen Brennan
Associate Professor of Education - Harvard Graduate School of Education
Explorations in Computational Tinkering

by

Carmelo Presicce

Signature redacted

Reader

Karen Wilkinson
Director of the Tinkering Studio - Exploratorium, San Francisco
Explorations in Computational Tinkering

by
Carmelo Presicce

Signature redacted

Reader

Sepandar Kamvar
LG Career Development Professor of Media arts and Sciences
Contents

Chapter 1 - Introduction 15
Computers to tinker with, contexts to tinker in 15
From computational thinking to computational tinkering 16
Physical and computational materials 17
Thesis overview 18

Chapter 2 - Background and related work 20
What we mean by learning 20
Constructionism 21
Creative Learning 23
Tinkering and tinkerability 24
Computational thinking and computational tinkering 25

Chapter 3 - Computational tinkering landscape 26
Scratch 26
Makey Makey 28
Arduino Paper Circuits 29
Programmable Sound Machines 31
Digital Lightplay 33
Discussion 36

Chapter 4 - Programmable Art Machines 37
From scribblings machines to programmable art 37
Activity structure and goals 40
Case studies 40
Case study 1 - Workshop in an after-school coding club 41
Case study 2 - Drop-in station at CoderDojo 46
Case study 3 - Free-play in the living room 52
Case study 4 - Workshop at Scratch coding club 58
Computational tinkering with LEGO Programmable Art Machines 63
Pseudo random scribblers and predictable turtles 63
Physical first (and foremost) 63
From controlling to programming 64
Chapter 5 - Looking back, looking forward

Reflecting on Learning through Computational Tinkering
  Learning by iterating 65
  Learning by exploring and experimenting 66
  Learning by reflecting 66
  Learning through a personal, social and emotional journey 67
  What is special about computational tinkering? 68

Future directions 69
  Making computation more accessible 69
  Bringing computational tinkering in schools 70
  A final note 71

Acknowledgements 73

References 75
## List of Figures

3.1 - The Scratch programming interface 27  
3.2 - A Play-doh game controller made with Makey Makey 28  
3.3 - The Arduino Paper Circuits kit and a collection of paper modules 30  
3.4 - Young girls creating a dance project with Scratch and Arduino Paper Circuits 30  
3.5 - A LEGO Sound Machine 32  
3.6 - A Programmable LEGO Sound Machine 32  
3.7 - An artistic installation created with Digital Lightplay 34  

4.1 - Examples of Scribbling Machines 38  
4.2 - A LEGO Art Machine 38  
4.3 - A programmable Art Machine and its programming script on the iPad interface 39  
4.4 - A programmable Art Machine used as a sample, and its related code 42  
4.5 - Sophie's attempt to make the machine move by installing the motor on the marker 48  
4.6 - Tom’s and Sophie’s Art Machines 49  
4.7 - Ray’s family playing with Art Machines in the living room 53  
4.8 - Ray debugging the Scratch code for his Art Machine 55
1. Introduction

Computers to tinker with, contexts to tinker in

In 1989, as a 6-year old, I got a very special gift from my parents: a personal computer. I remember how exciting it was to unpack the big boxes and getting ready to play with it. There was only one problem: I had no idea of how to make it work. And also, I didn’t really know what I could do with my new and expensive toy. All I got was a cryptic DOS manual I couldn’t even read, and the blind trust of my parents.

I learned pretty soon that I had to figure it out by myself, and that turned out to be the aspect I enjoyed the most. “Where should I connect this cable? What does this button do?”. The only way to discover was to try.

Tinkering has been the way I engaged with computers since my very first experience. Figuring things out by myself gave me a sense of ownership and confidence in my ability to understand without being taught. I got curious about new technologies and, more importantly, I learned that I could figure out many other things in the same playful and exploratory way.

Unfortunately, many people experience computers in a very different way. When a friend of mine got his first computer in high school, it was considered almost a sacred object, and his parents’ recommendation was: “Don’t break it!”. Needless to say, my friend didn’t get very engaged, and he started to see technology as something only for “experts”.

The reasons why my friend and I felt so differently about computers are not to be found in the technology. If anything, my friend’s computer was far more powerful, versatile, and easy to use than my first computer. The distinguishing element that led to my playful tinkering on one side, and his fearful reverence on the other, lay in the context in which the computer had landed.

In this thesis, rather than focusing on the design of tinkerable technologies, I will explore how to design contexts that allow and encourage tinkering with computers. More specifically, I will describe the design and facilitation of activities in which children can learn with computers through a playful, iterative, and fearless exploration.
From computational thinking to computational tinkering

When I heard the phrase "computational tinkering" pronounced for the first time by Karen Wilkinson, I immediately recognized the expression as a wonderful medium to communicate a powerful idea. My reaction was: "Whatever it means, I want to work on it!".

The pun is easily recognizable by researchers and practitioners in the field of education who have been exposed, in recent years, to the pervasive narrative of computational thinking. Computational tinkering offers a response to that narrative and, in the spirit of tinkering, it is both whimsical and very serious at the same time.

Teaching computational thinking is one of the main motivations to introduce coding classes for students of all ages, all over the world, since primary school. Although different definitions exist, computational thinking is commonly interpreted as the ability to apply computer science concepts to problem solving, with an emphasis on logic, abstraction, decomposition, analytical reasoning, and systematic planning.

Even though it is certainly important for everyone to learn how to use logic to solve problems, this perspective appears to be pretty narrow, as it gives little weight to other opportunities that come from engaging with computation - opportunities for creative expression and playful invention. In addition to the narrow scope of the expected outcomes, the dominant narrative of computational thinking also tends to suggest that a linear and didactic approach is the most appropriate way to introduce students to computation.

Computational tinkering extends computational thinking in both ways. From one side, it expands the range of desirable outcomes, to also include the ability to generate ideas, remixing other people’s work, design iteratively, develop a sensibility toward aesthetics, being able to question and accept feedback from others. From a pedagogical perspective, computational tinkering proposes a different approach to introducing computational concepts and practices to young people - one that consists in creating the conditions for a creative, playful, experimental, and iterative style of engagement with computational ideas and materials.

Obviously, tinkering is not opposed to thinking. Tinkering is a way of thinking, making, and interacting with phenomena. It's both a pedagogy and a playful disposition in approaching a problem or designing an artifact, that follows an explorative process guided by whim, imagination, and curiosity. It doesn't exclude logical reasoning and problem solving but it also encompasses creativity and self-expression; it doesn't exclude top-down design, but it also encourages bottom-up approaches; it doesn't dismiss abstraction, but it also values concrete ways of thinking, and concrete objects to think with. Ultimately, computational tinkering aims to help young people to engage with computational concepts, practices, and perspectives.
Sometimes computational thinking is summarized as: “thinking like a computer scientist”. I consider this perspective problematic for different reasons. First of all, it seems to suggest that a certain category of people, based on their profession, think “better” than others - and that everyone should learn to think like them. Also, most of the people think that computer scientists think in a very analytical way, carefully planning ahead and always sticking to their plan, but this is not necessarily true.

If we accept that computational thinking means to think like a (fictionalized) computer scientist, then computational tinkering means to think and act like an artist - or like most people think artists think and act - with a mix of play and inquiry, carefully and curiously noticing and then taking action based on what was noticed. In Epistemological Pluralism, Sherry Turkle and Seymour Papert (1990, p. 169) wonderfully illustrate the tinkering style of engagement with a similar metaphor: “The bricoleur resembles the painter who stands back between brushstrokes, looks at the canvas, and only after this contemplation, decides what to do next.”

Contexts for computational tinkering should not prevent planning or other styles of learning and thinking. On the contrary, as educators we should provide space and opportunities for people to think like computer scientist, and to think like artists. In this sense, the Programmable Art Machines activity that I describe in Chapter 4 is an attempt to acknowledge and encourage both ways of thinking. By describing the learning activities I designed and facilitated for this thesis, I hope to show that, when we provide space for tinkering, we also create opportunities for learners to express themselves through a diversity of styles of engagement.

Physical and computational materials

Tinkering involves putting things together and taking them apart, trying different things to see what happens, diving into building something, without having a clear idea of what it will end up being, or what to do next. Tinkering is an attitude, it’s a playful style of engagement with materials.

Some materials are particularly suitable for tinkering. Physical materials like cardboard, clay, pipe cleaners, or LEGO bricks, are relatively cheap and flexible, and they can be easily combined together and disassembled, and they offer endless opportunities for creation.

Computers are also tinkerable materials. Everyone who has created a computer program, knows that you can tinker with code. It is very rare to get a program to run correctly the first time. Software developers are used to tinkering with code: combining different instructions or
modules together, taking them apart or refactoring parts of the software, trying different ways to obtain the desired effect, iterating over and over.

Visual programming languages like Scratch are designed to make this tinkering process even more explicit and natural. For example, representing instructions as blocks and making them connectable like LEGO bricks, makes it easier to think about computer instructions as materials to build with.

Using a combination of physical and computational materials can open up opportunities for tinkering in new ways. From one side, being in a context in which people tinker with physical materials could inspire a more playful approach in building with code. On the other side, computational materials unique affordances can also inspire new types of artifacts that can be created in the physical world. In my explorations of computational tinkering I will look at the combination of digital and physical materials, providing contexts in which they can be blended together to provide new possibilities for playful experimentation and invention.

**Thesis overview**

This thesis is about tinkering, but it is also the result of my own tinkering with the idea of computational tinkering. It's the documentation of my personal, iterative, and playful journey in search for a meaning, and also an attempt to contribute to a larger conversation about designing and facilitating activities that bring together the capabilities of computation and the spirit of tinkering. In this sense, this thesis is an object to think (and tinker) with, and my explorations are an early attempt to understand the space of possibilities, opportunities and challenges.

In Chapter 2, I describe the historical and epistemological background in which this work situates, focusing on the influence of Constructionism and Creative Learning, and the importance of providing opportunities for learning through tinkering.

In Chapter 3, I give a broad overview of toolkits and activities I've been facilitating, and I reflect on how they relate to the idea of computational tinkering.

Then, in Chapter 4, I dive deeper into a particular activity developed around a specific toolkit, the Programmable LEGO Art Machines. I use a case study approach to present and analyze some workshops I designed and facilitated for children. For each of them, I share my observations and reflections in order to iteratively improve the design of the next one, and to get some insights on how people learn through computational tinkering. In the last case study, I focus more on how children see themselves in relation to tinkering and computation, sharing and reflecting on some of their answers to an interview.
Finally, in Chapter 5, I reflect on my experience, identifying some key characteristics of learning through computational tinkering and proposing directions for further work in the design and outreach of computational tinkering activities and ideas.
2. Background and related work

In this chapter I provide an overview of the cultural and epistemological context in which this work situates. Then, I review some interpretations of computational thinking and relate them to the emerging idea of computational tinkering. In the last section, I refer to influential work around the design of tinkering experiences and tinkerable computational toolkits.

What we mean by learning

The title of this section is inspired by a booklet by Aaron Falbel and Edith Ackermann (1998). In that work, the authors refer to learning as a natural process of "making sense of the world". In their view, learning occurs at any given moment of our lives, and not just when we are in a classroom, or when someone is teaching us something.

The idea of learning as a natural, active process, and the crucial role of the experience in that process, is not new. Many educators like Frobel (1885), Dewey (1938), Montessori (1949), and Freire (1970), have recognized the importance of learners’ agency and critical thinking, emphasizing the limits of learning models based on transmission of information. The constructivist theory of Piaget (1969) describes the learner as an active producer of knowledge, continuously adapting inner mental models to make sense of new experience. Papert’s Constructionism (1991) builds on Piaget’s Constructivism, to emphasize that making sense in our head is particularly well supported by making something in the world.

It is no coincidence that the research group that carries on these ideas, and where I have the privilege to play and learn, is named “Lifelong Kindergarten”. Mitchel Resnick, who leads the group, is inspired by how children learn in kindergartens (Resnick, 2007), through playful experimentation with materials and with ideas. That’s why the technologies and activities developed by the group are always oriented towards providing kids with opportunities for playful exploration and invention.
Constructionism

The assumptions and powerful ideas that constitute the foundations of this thesis are strongly influenced by two learning theories, Constructivism and Constructionism.

"Piaget’s Constructivism, Papert’s Constructionism: what’s the difference?". This is the question that Edith Ackermann answers in a work (Ackermann, 2001) that constitutes a beautiful exploration of the relationship between the two theories. While Ackermann’s original perspective wonderfully illuminates the similarities in the goals and differences in the approaches, in this section I offer a brief synthesis of the two theories.

The basic idea of Piaget’s Constructivism is that knowledge can’t just be transferred by one person to another. On the contrary, people actively create knowledge in their own minds, through a process of adapting existing mental models (schemas) to new information or experiences. This view implies an active role of the learner in building knowledge, and suggests that the process of learning is an active process of sense-making.

Seymour Papert, who worked with Piaget, embraced the constructivist learning model, and extended this theory, noticing how the construction of knowledge in our minds happens particularly well when we are actively involved in constructing something in the world. To put in his own words (Papert, 1991, p.1):

“Constructionism - the N word as opposed to the V word - shares constructivism's connotation of learning as building knowledge structures’ irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it’s a sand castle on the beach or a theory of the universe.”

Papert highlights the importance of contexts for learning. For a designer of learning activities, the word context includes an incredibly wide variety of elements, from time to space, social and cultural norms, available materials and technologies, structure of the activity and so on. The phrase consciously engaged suggests the idea of the learner being aware of her own process of learning. Creating opportunities for reflection about the creative process itself constitutes a key element that distinguish a meaningful constructionist activity from a process of random trials and errors.

Also, when the result of construction process is a public entity, the artifact becomes both personally and socially meaningful. Both these qualities include an affective, rather than a purely cognitive dimension. Finally, Papert points out that the outcome of the constructionist learning process can assume different forms and be expressed through different mediums, more or less tangible, yet always concrete.
Computers as objects to think with

A significant part of Papert's work has been focused on the role that computers can play in engaging children in learning through creating. The LOGO programming language he developed in the 60's, served as a powerful environment to enable kids to use computers as materials (Franz & Papert, 1988). In the first implementations of LOGO, the programming language was used to control the movement of a physical robot that resembled a turtle and could draw on the floor.

The tools and activities described later in this work also explore the relationship between physical and computational material in the same spirit. Computer instructions are the computational building blocks that can be used to create behaviors in the physical world.

As I will show in Chapter 4, today the equivalent of the turtle can be built by kids - for example using LEGO motors and sensors - and it can be programmed with Scratch, a modern programming language that has been developed in the same spirit of LOGO.

Planning and tinkering

Another fundamental contribution of Papert's work, extremely influential on this thesis, is the recognition and support for different ways of thinking and learning. In their work “Epistemological Pluralism and the Revaluation of the Concrete” (Turkle & Papert, 1990), the authors point out that the experience of programming, and more generally of creating, can assume different forms for different people.

Some people act more like "planners", who favor a top-down, divide-and-conquer approach: for them, the creative process consists in making a plan, dividing the problem in smaller parts and designing solutions for each of them. Other people are more similar to “bricoleurs”, or tinkerers, who favor a bottom-up way of constructing, engaging in a process that looks more like a continuous conversation with the materials, and refinement of ideas and goals while making.

Papert is far from claiming the superiority of one style over the other, but he observes (Papert, 1993, p.146) that:

“Our intellectual culture has traditionally been so dominated by the identification of good thinking with abstract thinking that the achievement of balance requires constantly being on the look-out for ways to reevaluate the concrete”.

Designing learning activities for computational tinkering is not in contrast with the idea of supporting computational planning. On the contrary, by providing space for tinkering, we offer multiple pathways into the learning experience.
Creative Learning

Building on Papert’s legacy, Mitchel Resnick, along with researchers in the Lifelong Kindergarten group, has developed a framework that highlights 4 key elements of a creative learning experience (Resnick, 2014).

These elements are referred as the 4P’s of Creative Learning:

- **Projects**: People learn better when they are actively engaged in making projects. The experience of coming up with ideas and bringing them into the world, designing, developing and testing prototypes, provides a rich context in which problems naturally arise, rather than being artificially imposed.

- **Passion**: When people are genuinely interested in what they are learning, they focus more easily and persist in the face of challenges.

- **Peers**: Learning is not a solitary activity, but it is often situated in a social context in which peers play a crucial role. Interactions among peers include collaborating, getting inspired and building on other people’s ideas, helping each other, sharing and exchanging feedback.

- **Play**: Creative learning involves playful experimentation. We learn more when we are open to exploring new ideas, taking intellectual risks, testing boundaries, trying new things and iterating.

The Creative Learning framework builds on the constructionist theory of learning and constitutes a useful lens to analyze and design creative learning technologies and activities.

Playful learning

Although all of the 4P’s are relevant in the design of creative learning activities, it’s important to clarify the meaning of “play” in this framework. The word play is undoubtedly overloaded with meaning, and it is commonly associated with the idea of having fun, enjoyment, and recreation.

In the context of Creative Learning, “play” refers more to a particular mindset rather than a certain type of activity. This mindset is best represented by the word “playfulness” or “playful learning” (Resnick, 2006): the attitude of being proactively engaged and curious, experimenting with ideas and materials without fearing mistakes.

Actually, in a tinkering context, mistakes are not only harmless but even desirable: they are rich opportunities for learning, as they provide new information about the phenomena or the
material. The active and engaged part of tinkering flips the narrative on making mistakes completely, since that’s what propels the investigation forward and motivates learners to keep making progress.

Also, a playful activity doesn’t have to be easy: as Papert describes with the idea of “hard fun” (Papert, 2002), people are more engaged by activities that are challenging, as long as they touch on their interests and passions.

**Tinkering and tinkerability**

Playfulness and experimentation are at the core of the process of “tinkering”. Although it is hard to crystallize it into a static definition, the essence of tinkering is well captured by the expression “thinking with your hands” (Sennett, 2008) while taking things apart and putting them together, playing around with tools and materials, testing boundaries and making mistakes in order to achieve a goal or to understand a phenomenon.

Martinez and Stager (2013) refer to tinkering as a mindset that involves a “playful approach to solving problems through direct experience, experimentation, and discovery”, while the term “making” involves a central role of the artifact. For Resnick and Rosenbaum (2013) making and tinkering are strongly interconnected: for them, “tinkering is a playful style of designing and making, where you constantly experiment, explore and try out ideas in the process of making something”.

Designing the conditions for learning through tinkering is not an easy task. Petrich et al. (2013) point out that in a tinkering experience there is “no set of instructions, and no prescribed endpoint.” That doesn’t mean that anything goes: on the contrary, that means that everything from tools and materials to facilitation needs to be curated very carefully. In their work, they also describe a rich set of design principles that encompass the design of the activity, the environment and facilitation strategies, and emphasize how crucial it is to take them in consideration “all at once”.

As computation introduces a new layer of expressiveness in tinkering experiences (Blikstein, 2013), researchers have been developing tools (Maloney et al., 2010) and design principles to promote tinkerability in computational construction kits (Resnick and Rosenbaum, 2013). Only little has been said about the context - activities and facilitation strategies - that can support tinkering across the digital and physical world. With the explorations in thesis I aim to contribute in that direction, looking at the design of tinkering experiences in computationally rich contexts.
Computational thinking and computational tinkering

Papert introduced the term "computational thinking" in Mindstorms (1980) describing a vision for learning environments where children could explore computational ideas while being engaged in creating meaningful projects. Computational thinking also appears in (Papert, 1996) where he argues that "the goal is to use computational thinking to forge ideas that are [...] more accessible and more powerful".

More recently, computational thinking has been popularized by Wing (2006) to indicate a skill set that emphasizes abstraction, decomposition, and pattern recognition, in order to find efficient ways to solve problems. Cuny et al. (2010) also describe computational thinking as the thought processes involved in problem solving, particularly referring to the ability of formulating problems in such a way that a computer can carry them on.

While the discussion around computational thinking is still open (Barr and Stephenson, 2011; Grover and Pea, 2013), the problem-centric approach has been welcomed by teachers (ISTE and CSTA, 2011) and influential organizations (Google, n.d.). Only a few researchers have been advocating for a broader definition of computational thinking that also involves practices and perspectives (Brennan and Resnick, 2012), that addresses inclusion and participation (Kafai, 2014), and that involves making projects in the physical world (Rode et al., 2015).

The approach of computational tinkering arises from the need to overcome the limitations of the current computational thinking narrative. Wilkinson et al. (2016) have recently proposed the term to describe "a playful approach to constructing with code". The idea is to extend concepts and practices valued by computational thinking - like analyzing, abstracting, and decomposing problems - with a constructionist approach that values generating and expressing new ideas, making personally meaningful projects, iterating, collaborating, and reflecting.

Although the ideas of computational tinkering have solid roots, the concept is still in its infancy. Defining computational tinkering is still a work in progress, and that makes this thesis work particularly challenging and exciting. What I will describe in the next chapter, and in the rest of this thesis, it's my personal - still evolving - perspective on what computational tinkering looks like.
3. Computational tinkering landscape

In this chapter I give an overview of activities and toolkits I've encountered and explored in my learning journey, illustrating how they relate to the idea of computational tinkering. Although this overview of the landscape is preliminary and not complete, it offers a sense of the variety of tools, materials, and media that can be used to support a playful exploration of computational ideas.

The computational tinkering activity landscape can be seen as a spectrum between two different perspectives or interpretations. In some of the activities I describe - like in the case of Scratch programming - computational tinkering means tinkering with code, while creating projects on a screen. With other activities - like Sound Machines and Digital Lightplay - tinkering refers mostly to the physical materials, while programmability helps expand the possibilities for expression.

In all cases, the affordances of the toolkit alone do not define the way in which people engage with it. It is how the activity is designed - in terms of prompts, activity structure, and facilitation - that can provide more or less space for people to tinker with computation or to leverage computation while tinkering.

Scratch

"Programming languages are the construction kits of the computational world" (Resnick & Silverman, 2005). I've been working for years as a software developer and I've been playing and building with code for a long time. But it was only when I tried Scratch (Resnick et al., 2009) - where computer instructions can be combined as if they were LEGO bricks - that I started seeing code as a material. The metaphor of snapping instruction blocks together was a powerful way to visualize how the lines of code I had composed for years were building blocks of computational artifacts. And it was only when I explored Scratch (scratch.mit.edu) with a 6-year-old in a CoderDojo club, that I realized how exciting and rewarding tinkering with code can be: trying different things, discovering new possibilities, and being surprised by what we create.
Scratch has been explicitly designed for tinkerability (Resnick & Rosenbaum, 2013): the programming environment makes it easy to explore functionalities and do quick experiments; it provides a rich set of instructions and carefully curated media libraries; and it encourages cross-pollination of ideas through the online community, and deep learning from interactions with peers — not from instructions of an instructor.

Even though Scratch supports tinkering, it doesn’t mean that everything people do with Scratch is tinkering. Sometimes Scratch is introduced in a very didactic way, through activity prompts that are narrow and directive. For example, asking students to replicate a particular project, or following a well-defined sequence of steps, inevitably limits the space for discovery and experimentation. The design of the activity can take tinkering out from a tinkerable tool. Instead, computational tinkering activities are open-ended, and should be designed to encourage people to explore and play with code.

At the same time, tinkering activities can - and should - also provide scaffolding. A good example is “animate your name”, one of the many activities included on the Things to Try Scratch webpage (scratch.mit.edu/go). In that activity, newcomers are encouraged to explore basic features of Scratch - playing with code to try different graphic effects or to move objects on the screen - in order to make the letters of their names interactive.

Usually, tinkering with Scratch happens on a screen, where computer instructions are used to manipulate digital media objects, like graphics or sounds. It’s important to highlight that the phrase “computational tinkering” doesn’t require physical materials: it refers to the exploratory
and playful style of engagement in creating with code, no matter if the artifacts resides in the physical or in the digital world.

Of course, Scratch can also be used as a computational support to enhance activities that involve tinkering with physical materials. One way to do that is to connect Scratch to external devices - such as Makey Makey, Arduino or LEGO WeDo - that act like gateways to the physical world. In the next sections I will describe some of these tools and activities, illustrating how Scratch can be used as a computational glue between the physical and computational domains.

Makey Makey

With Makey Makey people can create unconventional input devices, using any objects that conduct electricity. The device behaves like a computer keyboard, but keystrokes can be generated by touching ordinary objects, instead of keyboard keys. Projects made with Makey Makey include game controllers and musical instruments made with materials like Play-Doh, metallic objects, or pieces of fruits.

![Fig. 3.2 - A Play-doh game controller made with a Makey Makey](image)

Like Scratch, the Makey Makey has also been designed to support tinkering (Resnick & Rosenbaum, 2013). It invites a playful experimentation with a potentially infinite range of everyday objects; as its inventors say, it encourages seeing “the [physical] world as a construction kit” (Silver, 2014). But the tool itself can’t do all the work, and sometimes people use Makey Makey in ways that are not exactly aligned with designers’ intentions. When I was giving a workshop using Makey Makey in China, one of the participants was extremely happy to experiment with different materials; he told me that he had previously participated in a workshop where more than 100 people were asked to make the same project: a piano made of bananas.
Obviously, that activity didn’t leave space for creativity and tinkering - not to mention the waste of bananas.

Designing an activity that supports tinkering requires a thoughtful selection of the materials offered as well as prompts that are wide enough. I had the opportunity to participate and to facilitate workshops with Eric Rosenbaum, one of the co-inventors of Makey Makey, where participants were asked to create bizarre game controllers. Some of the prompts included making a game controller that involved water, or that required intense physical exercise, or that could be played only with feet, and so on. In that case, constraints were used to unleash creativity and encourage experimentation in the physical world.

Experimenting with everyday objects to create a game controller certainly involves tinkering in the physical world, and it somehow requires some computation, both from the microcontroller on the Makey Makey board, and from the computer where the videogame is running. Nevertheless, I wouldn’t consider it a “computational tinkering” activity unless it involved using code to create the actual videogame. In my view, it’s only when the Makey Makey is used in combination with a programming language like Scratch that it can unleash its full potential as a computational tinkering toolkit.

Arduino Paper Circuits

Makers around the world use Arduino to create almost anything that involves electronics, from simple blinking lights to robots, quadcopters, and 3d printers. Even though Arduino has made electronics more accessible and tinkerable, it still presents obstacles. Creating circuits on a breadboard is not very intuitive; the programming language not easy to grasp; and the process of uploading the code on the microcontroller prevents having immediate feedback.

That’s why people at the Tinkering Studio have devised a setup that makes experimenting with Arduino more immediate and intuitive (Jenkins & Catrett, 2016). Their toolkit consists of a set of basic electronic components - like LEDs, switches, and sensors - that are mounted on colorful squares of paper. These modules can be easily connected to each other and to the Arduino microcontroller using alligator clips. To make this connection easy, the Arduino is mounted on a wooden block, and its pins are electrically connected to copper nails where the alligator clips can be easily attached. Also, they leveraged a Scratch extension (Hanning, 2017) to control the Arduino board in real time, using Scratch programming blocks to control inputs and outputs. For example, it is possible to create a simple script that makes an LED light blink every time a “paper button” is pressed.
I've been using this toolkit to facilitate workshops with children and with educators. After demonstrating how to create and program a simple circuit, I encourage participants to experiment with paper buttons and LED-light paper modules in Scratch. One of the projects that three young girls created was a “dance party”: pressing the paper switch activated colored LED lights, and also triggered music and animated dancing characters on the computer screen. In the process, they tinkered with Scratch code, with different kind of media on the screen, and with electrical connections in the physical world.
The combination of Arduino Paper Circuits and Scratch is an interesting tool for exploring computational tinkering, across both physical and digital worlds. The alligator clip connection is crucial to simplify the process of creating an electrical circuit; also, the pre-built paper modules provide an easy starting point for experimenting with circuits, and suggest the integration of electrical components into custom paper crafts.

The use of Scratch rather than the standard text-based Arduino environment brings advantages in terms of tinkerability. While the traditional way of programming Arduino involves editing the code, compiling it and uploading it on the board, with Scratch any change in the code has an immediate effect. This reduces to zero the time between trying something new and seeing the result, enabling fast iterations and quick feedback loops, that are vital to the tinkering process. Also, Scratch provides a stage on the screen that can be used in different ways. The screen can be used to create animations that complements the physical artifact, or videogames that can be controlled using the paper-made switch buttons.

One of the main challenges of this toolkit, though, is that it requires a certain degree of knowledge - or adequate scaffolding - about how electrical circuits work and how components can be connected to Arduino. If different components are randomly connected to each other, it’s easy to come up with circuits that are difficult to understand and debug, and sometimes, as in the case of short circuits, potentially dangerous.

Programmable Sound Machines

With LEGO Sound Machines, people can use LEGO pieces - like gears, axles, or lift arms - in combination with everyday objects to create musical contraptions. This activity is part of ongoing experimentation led by the Tinkering Studio and the LEGO Idea Studio, and involves many other tinkerers worldwide, who get inspired and contribute remotely, sharing ideas on social networks as Twitter, using the hashtag #LEGOtinkering.

The main support is typically a pegboard where a LEGO motor is installed, and where other LEGO pieces can be assembled to create gear sets, linkages, and other contraptions. The pegboard was created out of a desire to tinker with LEGO configurations systematically on a vertical axis. Many other objects can be connected, including egg shakers, metallic parts like xylophone bars, small and big percussion instruments, and so on. When the motor spins, it originates a sound pattern that is defined by the particular configuration of the mechanical parts and the computational structures.

Sound machines can be programmed using Scratch and a LEGO WeDo kit. Instructions in Scratch enable control of the speed and direction of the motor, and also interaction with the WeDo motion sensor and tilt sensor.
Fig. 3.5 - A LEGO Sound Machine

Fig. 3.6 - A Programmable LEGO Sound Machine
Part of the tinkerability of the Programmable Sound Machines comes from the tinkerability of the physical and the computational materials. Assembling and disassembling LEGO parts is extremely easy, and all the possible combinations are pretty much infinite. It’s easy to start with a simple mechanism, but the exploration is wide open: there are no limits to the type of everyday objects that can be used to produce sounds, and finding ways to connect them to LEGO structures also involves a certain amount of tinkering.

From the computational side, Scratch scripts interact in real time with the physical construction and can be modified even while the sound machine is playing. The programming language permits the creation of loops and delays, which make it interesting to play with timing. In addition to the sounds produced by physical objects, other sounds played by the device can also be integrated, becoming part of the musical composition. Finally, the available sensors can be used to make the sound machines interactive: for example, changing the tempo according to the inclination of a tilt sensor, or depending on the distance of the player’s hand from the motion sensor.

I helped with the facilitation of a Programmable Sound Machine activity organized at MIT Media Lab by the Tinkering Studio during an event in memory of Seymour Papert. The activity was designed to engage participants in a casual way: every table in the room had a pegboard with a programmable motor and some pre-built contraption, one tilt or distance sensor, some additional materials, and an iPad running Scratch. Some people got extremely involved in tinkering with gears and linkages. A few others experimented with programming, playing with the speed and direction of the motor, and with different timing. Most of the participants were looking curiously at (and listening to) existing machines, often changing something here and there, while casually chatting around the table.

Some of the challenges of the Sound Machines activity were related to the limited capabilities of the programming language. For example, compared to the Scratch WeDo extension, the iPad Scratch prototype only allows for three different motor speeds (instead of a scale from 0 to 100), it can only detect when an object is waved in front of the distance sensor (instead of having a value between 0 and 100), and can only detect if the tilt sensor was inclined (instead of getting an indication of the angle). Also, the Scratch prototype on iPad doesn’t yet support if-else statements, or other control structures, so the limited expressiveness of the programming language makes it less compelling to tinker with code, rather than to experiment in the physical world.

Digital Lightplay

Lightplay is an activity developed by the Tinkering Studio at the Exploratorium, in which light is both a phenomenon to investigate and a medium for artistic expression. People leverage shadows, reflections, and colors to create eye-catching artistic installations and vignettes.
The traditional kit consists of a set of colored light sources that can be easily positioned and oriented toward physical objects to create interesting light effects. The kit also includes a motorized turntable on which objects can be placed, so that their motion can create dynamic lights and shadows.

The digital version of Lightplay, developed by the Lifelong Kindergarten Group, also includes an electronic board that allows up to three light sources and one turntable to be programmed on a computer or a tablet using Scratch. Composing instruction blocks on the screen allows one to create programs to change the color of the lights, control the speed and direction of the turntable, and interact with external sensors.

Digital Lightplay supports tinkerability in many ways. The effects of experiments in the physical and in the programming environments are immediately visible. This is pretty obvious in the physical world: when people change the position or orientation of the lights, the resulting effect changes in real time. From the computational point of view, the programming environment inherits the tinkerability of Scratch: programming blocks provide immediate feedback and they blink while they are executed, making the program flow visible and easier to debug.
Lightplay supports the exploration of familiar objects in unfamiliar ways: doilies, plastic meshes, lenses, prisms, plastic glasses - all provide opportunities for a playful investigation of light, shadow, reflection, and motion. The computational aspects of Lightplay open up new possibilities: the ability to program lights and motion over time presents opportunities for storytelling; sensors can help creating interactive narratives; other media, like sounds and music, or graphic objects on a screen, can also be integrated as part of the vignette.

The Lightplay activity developed by the Tinkering Studio is usually divided in three different phases. In the first part of the activity, participants are encouraged to explore and experiment with different materials, to develop a sense of what it is possible. In the second part, they work on creating their own vignette. Finally, they share their work with other participants and reflect on their experience.

In the first, explorative part, physical materials are distributed on three different stations, each focused on investigating a particular phenomenon: shadows, reflections, and colors. With Digital Lightplay, an additional station offers the possibility to explore the computational material. Participants get the chance to assist a quick demonstration of how to use the programming language and they start experimenting by themselves.

I've been co-facilitating the Digital Lightplay activity in a variety of contexts, with people of different ages, including students and educators with different familiarity with Scratch. In my personal experience, people - including those with experience and familiarity with programming - have been generally spending most of their time tinkering with physical materials, while the computational aspects have been usually overlooked. Sometimes the programming interface has been used to set the colors of the lights once and for all, and even people who experimented more with code, have generally created scripts that showed little intentionality or a lack of understanding of the programming instructions.

The Digital Lightplay experience seems to be unbalanced toward tinkering in physical world. Physical and computational materials have different affordances, and the introduction of programmability into an activity designed for physical tinkering poses interesting challenges. How can we remix or redesign the activity to make computation more meaningful, compelling, and intuitive? Or is it necessary in the first place?

I discuss these questions in the final chapter, after describing case studies that involve Programmable Art Machines, another activity developed by the Tinkering Studio, and computationally enhanced by the Lifelong Kindergarten. For now, it is worth pointing out that the design of some computational tinkering activities raise similar questions and share similar challenges.
Discussion

Even though this overview includes only a few examples, the tools and activities I've described above should help create an early map of the computational tinkering space, and hopefully clarify my perspective on computational tinkering.

From one side, computational tinkering means "promoting tinkering in computational activities". On the other side, it means "integrating computation into physical tinkering activities". Both these perspectives are equally valid, and both these aspects - even though with different weights - are present in each activity I described. Tinkering with Scratch code on a screen always involves manipulating media objects, and tinkering with physical objects in Digital Lightplay always involve a little bit of programming.

In the next chapter, I analyze more in depth another activity - Programmable Art Machines - that falls more on the physical-tinkering side of this spectrum. In addition to presenting the toolkit and the activity, I describe and analyze concrete use cases. Through the empirical study on Programmable Art Machines, I try to unfold the challenges of introducing programmability in a context in which tinkering with physical materials has a central role.
4. Programmable Art Machines

It's a Saturday afternoon in Bologna, Italy. In a study room of the University, about 20 children are sitting in front of their computers, combining together Scratch programming blocks to animate objects on the screen. In another room, downstairs, other children are combining together yogurt containers, colored markers, and electrical motors to animate creatures that draw on the floor. They are all participating to the same event - one of the many CoderDojo meetings where kids learn how to create with technologies - but they seem engaged in different types of activity. In one room, kids are coding with Scratch; in the other one, kids are tinkering with scribbling machines.

That dojo is a special one. We have called it MITic dojo, to celebrate two very special guests, visiting from MIT: Mitchel Resnick and Natalie Rusk. While I’m frantically running up and down the stairs between the rooms, to enjoy watching children’s projects coming to life, and to make sure that kids, parents, volunteers, and our special guests are having a good time, I keep thinking: does it make sense to keep coding and tinkering as different activities? How can we bring the spirit of tinkering and the capabilities of computation together in one room?

About two years later, I’m a student at MIT, and I’m still looking for answers. Mitch and Natalie are now my mentors, and we are working closely with researchers at Tinkering Studio, Reggio Children, and LEGO Foundation, to bring computation and tinkering together in one room.

In this chapter, I describe the genesis of the Programmable LEGO Art Machines toolkit, and I explain why I choose it to conduct my deeper investigations on computational tinkering. After presenting the general structure of the activity I designed, I share a few stories of young learners who have been engaging with the toolkit and some reflections about design and facilitation challenges I’ve been facing along the way.

From scribblings machines to programmable art

Researchers at the Tinkering Studio have been making - and helping people make - scribbling machines for a long time. Scribbling machines, as the name suggests, are machines that make scribbles. They are made of recycled containers, old CD-ROMs, strawberry baskets, popsicle sticks, and all sorts of random objects you can think of. All they have in common is a small electric motor, a battery, and some colored markers drawing on the floor.
The motor is usually attached to an offset weight, so that its vibration makes the machine move in an unpredictable way, leaving colored traces on the floor. Every little change in the structure - even the ones that are hard to notice - can strongly affect the way it moves, so the resulting scribblings are always a surprise.

What happens when you build a scribbling machine out of LEGO bricks? Researchers at the Tinkering Studio, in collaboration with the LEGO Idea Studio, made some experiments, observing how the affordances of LEGO materials affect the building process and the aesthetics of the drawings that these machines are able to generate (The Tinkering Studio, 2016). Then, they put together a set of LEGO Technic pieces, LEGO motors, and battery packs, and sent them to 30 museums and learning spaces all around the world, inviting educators and researchers to try them out with their audiences.
One of the kits ended up on my desk, and as soon as I opened the box I couldn’t wait to play with it. After a few minutes, a simple machine was running on the table and I had to chase it to turn it off, pushing the switch button on the battery pack. While I was trying to control the unstoppable machine, I realized that I could tame it using the Scratch programming language.

This episode also reminded me of a comment that some 3rd grader shared when I was visiting the primary school at the Malaguzzi International Center in Reggio Emilia. They had created many beautiful scribbling machines but they still had a problem: “These robots go wherever they want. Now we are working on making them go where we want”.

To make my Art Machine go where I wanted, I could simply replace the original LEGO motors and battery packs with programmable LEGO WeDo motors. Also, the LEGO WeDo kit includes a motion sensor and a tilt sensor that I could use to make the scribbling machine more interactive.

The LEGO WeDo based machines can be controlled remotely via bluetooth using a Scratch extension: in addition to the standard Scratch blocks, that are used to create animations on the computer screen, additional blocks are available to control the speed and direction of the motor, detect changes in the sensors, and so on.

In addition to the Scratch extension, the LEGO WeDo kit can also be programmed using a prototype of Scratch 3.0 that runs on iPad devices. The iPad prototype has a reduced set of programming blocks and doesn’t allow to create animations on the screen. Experimenting with the new form of programming in the physical world, using a minimal set of instructions, was another reason that made me excited to try the activity with kids.
Activity structure and goals

In the design of the LEGO Art Machine activity, I focused on learners’ first experience. The goal was to spark curiosity and to support children’s explorations of the provided physical and digital materials, encouraging them to create a personal artifact and to share their reflection about their process. Therefore, I designed a self-contained, semi-structured experience, that could be seen as a one-time workshop, or as the first step of a series of activities. With the help of my team of collaborators, I personally facilitated the Programmable Art Machines workshop in a variety of locations and contexts, including different after-school coding programs, a youth center, and a family's living room.

Although the available time and the space constraints differed between locations, the overall structure followed a similar schema:

- Introductions / icebreaker
- Presentation of samples and materials
- Building activity - alone or in pairs
- Sharing and reflections

According to the context, this overall structure could be more or less explicit. For example, in the context of a 2-hour workshop, I planned a certain amount of time for each phase. Other times, like in a drop-in tinkering station or in a free-play session at home, the structure was less visible and the time management more flexible.

The goal was always to provide opportunities for kids to engage in a playful, iterative, and exploratory process of creation using physical and computational materials.

Case studies

In the following case studies, I illustrate how I designed and facilitated 4 different Art Machines workshops with children. For each case, I describe my research goals, I share my observations about the experience (including a detailed learning story) and I reflect on those observations in order to iteratively improve the design of the next workshop and also to get some insights on how people learn through computational tinkering. In the last case study, I focus more on how children see themselves in relation to tinkering and computation, sharing and reflecting on some of their answers to an interview.
Case study 1 - Workshop in an after-school coding club

Workshop overview

Research goals

This workshop was my first opportunity to playtest the activity with children. I had explored the toolkit by myself and I had played with it with my interns before, but I had no idea about how kids would react, so I approached this first iteration with a very open mind.

The main goal was to address some preliminary questions around engagement and the creative process across the physical and digital worlds. What will the children like about the activity? Will they spend more time exploring and building with LEGO bricks or with code? What kind of artifacts will they create, and how will they take advantage of computation? Are they going to explore and iterate?

Also, I wanted to identify challenges about the activity structure, the environmental design, and facilitation strategies, in order to inform and improve the design of the next workshops.

Setting and participants

The setting for this activity was an after-school coding club for girls, hosted in an elementary school in the Boston area. The 12 participants were 3rd and 4th grader girls, they had some experience with Scratch but they were not familiar with the iPad prototype. The facilitation team included me, their teacher, and another educator. Neither of them had ever facilitated the activity before, but they were familiar with Scratch and with the LEGO WeDo kit.

The time constraints were very strict. We only had 1 hour for the entire activity, which included: introducing the activity, presenting a sample model and the material, working in pairs, sharing the drawing machines in a collective performance on the floor, sharing reflections / suggestions as a big group.

The activity was hosted in one of the school's technology classrooms. Instead of using the lab's desktop computers, we decided to use iPads to leave the children free to move, and choose the place where they wanted to work. We used a carpet on the floor as a central place to make the materials easily accessible to everyone. In addition to the LEGO pieces, also craft material was available, including pipe-cleaners, pom poms, googly eyes, and colored tape. Big sheets of paper were also used as a canvas for the drawing machines.
Observations

Workshop description

After welcoming participants, we started the activity sitting on the floor. I showed children a pre-built model and a simple script to control it. The model used an offset weight mounted on the motor, two markers, and a motion sensor. The script on the iPad consisted of three blocks and generated an intriguing behavior: when someone waved their hand (or any object) in front of the motion sensor, the motor turned in one direction for a second, in the opposite direction for another second, and then stopped. Children immediately engaged in interacting with the machine, waving their hands to activate it, and with the iPad program, tapping on the screen to activate the motor.

After playing briefly with the model, every pair of students got a WeDo kit and an iPad, and they were encouraged to create their own drawing machine. Most of the participants started by exploring the available LEGO pieces, then collected some material and naturally spread in different places of the room, going back and forth to gather additional material or to look again at the sample when needed.

All the groups immediately engaged in building some sort of physical structure around the WeDo hub, and they started decorating them early on, using the available craft materials. Many of them struggled with positioning the motor, and spent a good amount of time before figuring out how to use the motor to move the machine. Only a few children experimented with programming at an early stage, and when only 10 minutes were left for the building activity, none of them had tried adding markers and testing their machine's drawings.

At that point, I decided to explicitly encourage them to try their machines even if they were not perfect yet. To make it easier, I set up a big sheet of paper on the floor, next to each group. After
that, the first scripts started popping on the iPad screens, and the first drawings appeared on the floor.

In the last part of the activity, kids brought all their Art Machines in a common “arena”. The arena was a great opportunity to share ideas and reflections. Kids were looking at other people’s machines, noticing intriguing behaviors, asking questions, and modifying their scripts on the fly. “Look, that’s my program!”, or “Wow, it’s changing color! How did you do that?”.

All the Art Machines shared a structure similar to the sample project we showed in the beginning, but each one had a different look and presented a distinctive feature. For example, a group had added a third marker, another used a leg-shaped piece instead of an offset weight for locomotion. Also, most of the drawings showed circular patterns, but each of them was unique. One machine was drawing a “palm tree” shape, as it was programmed to periodically switch the motor direction back and forth.

Most of the kids’ comments were related to the experience of building in the physical world. Some shared that the experience was hard but they really enjoyed it, others were suggesting technical improvements, like adding the possibility of controlling two motors, or making the structures easier to build.

Even though as a facilitator I tried to follow and support all the different groups, jumping from one to another, in the section that follows, I describe more in detail the learning story of two of the participants, Emma and Jenny, and their attempt to build an ambitious project. Looking at the workshop from their perspective can help to unpack how the tinkering process looks, in order to better support it through the design and facilitation of the activity.

Learning story - A challenging journey (making a LEGO car)

After looking at the sample and exploring the available materials on the floor, Emma and Jenny had an idea: “Let’s make a car!”. They collected some LEGO pieces to build a frame structure, and tried different solutions, combining bricks together and taking them apart. At some point, they realized they could use 4 big gears as wheels: they connected them to the structure, and tested them out, moving back and forth the car with their hands to see if the gears had enough grip.

Then, their explorations turned into the aesthetic domain: it was time to make their creation look beautiful. “Which one will be the front?”, “What if we put some pom-poms here?”, “What can we use to stick the googly eyes on?”. This kind of questions guided their experiments, and it took them a good amount of time, and tinkering, to reach a point in which both of them were satisfied with how their car looked like.

But when it looked like their machine was almost ready, Emma and Jenny realized they were stuck. They hadn’t considered how to attach the programmable motor to the rest of the
structure, and now that it was decorated, it was harder to modify. Also, how could a single motor drive 4 wheels? Transferring the motion from the motor to the rest of the car was a big challenge, especially because they didn’t have familiarity with LEGO parts and gears. They tried different solutions, but nothing seemed to be working: the motor was spinning, but the machine wasn’t moving. When only 10 minutes were left before the final sharing party, they decided to go for a plan B.

They got back to the materials station, this time to analyze the sample model. After observing it carefully, they got some LEGO bricks and started building a similar model by themselves. Encouraged by the facilitators, they started testing their model more often. As soon as the motor was in place and connected to a leg, they wrote a simple script and tried it out, happily verifying that it was working. So, they added the first marker and tried again, as they cheerfully watched their machine making the first signs on the paper. Then, they made some adjustments, added a second marker, tried again, decorated with a pom-pom and googly eyes recycled from the previous model, and in few minutes the new prototype was ready to showcase in the shared arena.

While their artifact was moving with the others, Jenny got intrigued by observing all the machines drawing together. Emma, instead, was focused on her machine and kept experimenting with the code on the iPad. “Now I want it to do something else,” she said, as she was adding more and more instructions and nested loops to the scripts. She enjoyed snapping blocks together, but the increased code complexity didn’t change the movement of the machine significantly, and it made it harder to understand it. On the other hand, tinkering with code allowed Emma to discover new functionalities, like how to randomly change the light color on the machine.

Analysis and reflections

This first workshop helped me to highlight three important challenges. First, the available palette of computational materials - i.e. the instruction blocks that could be used - seemed to be less engaging than physical one. Second, iterating between the physical and computational world needed scaffolding and encouragement. Finally, the variety of the projects was pretty limited and suggested a more careful design of the sample projects.

Engaging with computational materials

Overall, kids seemed particularly attracted by building in the physical world with the LEGO and craft materials we provided. They spent the majority of their time building and tweaking their artifacts, and got interested in the coding aspect only towards the end of the activity. For example, Emma and Jenny spent a lot of energy trying to build their car structure, and only started experimenting with code very late. Obviously, kids had to build something before they
could program it, but most of them were waiting to perfect the structure before even trying to make it move.

Also, understanding the programming interface proved to be more challenging than expected. Some of the kids only used the iPad interface to switch the motor on and off. Others, like Emma, experimented more with code; but even when they created complex scripts, they generally looked like a random combinations of instructions, rather than the result of an intentional process. In any case, as soon as the motor was spinning, children moved their attention to the physical machine.

On the other side, kids enjoyed controlling the machine remotely and it was interesting that some of them played with the code while their Art Machine was moving in the arena. For example, Emma specifically said: “Now I wanna do something else”, keeping the Art Machine as it was, and playing with the scripts on the iPad to change its behaviour. She understood the potential of shaping the drawings making changes in the script, but it was hard for her to understand how, especially when the code was overly complex.

**Supporting iterative design**

One of the main challenges of this workshop was to encourage and support an iterative style of engagement, especially because the time was very limited and the iteration cycles had to be short. Children continuously tried new things and improved their projects, but the scope of their iterations was often limited to a specific phase, typically building the LEGO structure, without trying to see how the machine would behave. It looked like there was an implicit assumption on how to proceed: first build the structure, then decorate it, then create a script, and finally test it out.

When we provided children with an accessible platform to test their machines - and some encouragement - they started iterating on the full cycle, using the machine drawings as a feedback to guide their design choices.

**Diversity of outcomes**

Another challenge was about supporting a broader diversity of projects. Many of the projects that kids created were variations of the sample model. Some of the children took intellectual risks, trying to design something different - like the steering car that Emma and Jenny wanted to build originally - but when they encountered obstacles, due to the complexity of the project and the scarcity of time, they felt back on replicating the sample project that was available.

It is very likely that providing only one sample project might have biased the type of machines that children designed. Also the attention we gave to the sample in the beginning of the activity may have played a role, making children think that it was the expected outcomes. Ultimately,
having more time to experiment would have certainly generated a wider diversity in the artifacts. These observations helped redesigning the next iteration of the workshop.

**Main insights and next step**

The main insights that came from this iteration can be summarized as follows:

- Children deeply engaged with the activity, even though the time was very limited
- Physical materials were more compelling, at least in the beginning
- The Scratch iPad programming interface is not very intuitive yet
- Iteration needs to be encouraged and supported
- Sample projects can have a big impact on diversity of outcome

In the workshop that followed, I took into account these observations. I designed the space and arrangement of the materials to make it easier to try things out and iterate, I provided more scaffolding for the exploration of the computational environment and I offered a larger number of sample projects.

**Case study 2 - Drop-in station at CoderDojo**

**Workshop overview**

**Research goals**

With this workshop I wanted to play with the structure of the activity, trying it in a drop-in setting. Kids were free to come and go whenever they wanted and to spend as much or as little time as they preferred. I also wanted to follow some participants' learning trajectories more closely, and encourage a closer feedback loop between building, coding, and generating the drawings.

In the design of the activity, I incorporated some changes based on the insights I got from the previous experience. To encourage more diversity, I provided more samples. To make the coding experience easier, I spent some time demonstrating the basics of the programming language. To encourage iterations, I made it easier for children to test the drawings by covering all the tables with paper.

The goal was to verify whether those changes would help children get more engaged with computation, iterate more effectively, and generate more diverse projects based on their interests and abilities.

**Setting and participants**
CoderDojo clubs are free coding clubs for kids, usually hosted during the weekends in locations like public libraries or tech-companies. As a volunteer in the CoderDojo Charles River in Boston, I proposed to set up a Programmable Art Machines table in the room where children - aged 6 to 12 - were programming with Scratch. Another CoderDojo mentor helped me with the facilitation.

The LEGO materials were arranged on a big table. I created three simple Art Machines models, and just left them moving and drawing on the table, along with the rest of the LEGO pieces and craft materials. To avoid influencing kids' creations, I didn't describe or explain how they were built: the purpose was mainly to spark curiosity and attract kids to the table. Also, the table was entirely covered by paper, so that kids could test their Art Machines directly on their working area.

**Observations**

**Workshop description**

As kids approached the table, I gave each of them a LEGO WeDo kit and an iPad, I showed them the basics of the programming environment and pointed them to the LEGO materials, as they soon started building their Art Machines. In order to provide more space for experimentation, I didn't explicitly define a prompt: children deduced the goal of the activity by just looking at the samples and at other participants' creations. Children were working at the same table facing each other all the time, so that the working area was the equivalent of the shared arena experimented in the first workshop.

Among the kids who attended the CoderDojo Scratch session, four of them showed interest and spent their time - about 2 hours - building and refining Art Machines.

Even though they spent a significant amount of time building the structures, kids engaged with computation right away and their machines started generating drawings early on. Their first prototypes were generally simple and, even though the scripts didn't change much, the Art Machine structures complexified over time.

This time, instead of spreading my attention among all the different projects, I decided to follow more closely one little girl's explorations, while Mark - the other facilitator - supported the other 3 kids. Part of the reason is that I thought she might have needed more help, due to her young age. More important, I was interested in observing the thought process of such a young child engaging with Art Machines.

**Learning story - The joy of tinkering**

Sophie is a 6-year-old little girl who loves to create colorful animations and funny stories with Scratch. She had attended the dojo a couple of times before, and she approached the Art
Machines table with Tom, her 9-year-old brother. While Tom was attracted by the possibility of building with LEGO bricks, one of his favorite activities at home, Sophie was fascinated by looking at the machines moving and leaving colorful marks on the table.

After a quick overview, I gave each of them an iPad and a LEGO kit, and decided to follow more closely Sophie’s explorations. Her first intuition was to connect the motor cable and a marker to the hub; but, to her surprise, nothing was moving. So I suggested to have a look at the iPad, and she started exploring the iPad Scratch interface, tapping on all the available instructions. After a few attempts, she figured out how to make the motor spin. We also added an axle to make the motor movement more visible, but even though the motor was making the axle spin, nothing else seemed to be moving.

As the machine wasn’t moving, Sophie thought that she could give the machine a hand, and she literally started moving the machine on the paper using her hands, drawing a heart and a handwritten message: “Hi”. I was fascinated by her creative human-assisted solution to the motion issue, but I wanted to see how far she could go, so I asked: “Wouldn’t it be nice if it was moving by itself?”. She agreed, and she started thinking about it, trying different ways to place the motor.

At some point, her eyes lit up as she had an intuition: she placed the motor on top of the marker using an axle and said: “The pen needs to move!”. So, connecting the motor to the pen looked like a very good idea, and she couldn’t wait to try it out.

![Fig. 4.5 - Sophie's attempt to make the machine move by installing the motor on the marker](image-url)

When she gave the command using the iPad, the motor started spinning until the cable was twisted around the marker and it couldn’t go any further. Even though it wasn’t exactly what she
was expecting, at least something was moving! She got really excited about the motor spinning, so we decided to keep that configuration and playing with coding.

Playing with the code, she realized she could make the motor spin in the opposite way to unwind the cable, and she was excited to make an incredibly long script in which the motor was spinning alternatively in one direction and then in the other. Sophie had learned the concept of sequentiality in a natural way and she was visibly enjoying using it to make the motor spin.

While it was fun to see the motor spinning, that didn't help with the machine movement, yet. So, after some time, I brought back the problem of locomotion and invited her to look at what her brother was working on, just next to her. After carefully observing her Tom's complex model, she built a similar leg, and tried to connect it in multiple ways until she successfully made her Art Machine move, applying the same concept to her simpler model.

When she ran the script and saw the machine moving in a crazy and unpredictable way, she was so excited that she couldn't wait to make her dad see her creation. "Daddy, daddy! Look at what I did!" she yelled enthusiastically. She was very happy and proud to share her accomplishment with her dad and brother, and she watched for a while her creature drawing on the table, before a well deserved break.

Analysis and reflections

Overall, the changes I introduced in the activity - bringing more samples, making it easier to test the drawing on the working table, and providing initial guidance with the programming interface - seemed to have a beneficial effect.

Also, Sophie's story showed how a computational tinkering experience can look, when adequately supported by a careful design of the activity and a close and attentive facilitation.
Earlier experiments, quicker iterations, more diverse explorations

Compared to the previous workshop, children managed to get their machines draw sooner. Having the table covered by paper and more models running around, certainly helped; Sophie, for example, couldn’t wait to have her machine drawing something, to the point that she moved it around with her hands, when she wasn’t able to make the motor spin.

Kids also iterated faster and through the entire process. They built simple prototypes that could draw something, and they gradually complexified their projects, starting from the observation of the machine movements and drawings, to iteratively changing the structure and - although less often - the code, over and over again.

I also observed a broader diversity in the outcomes. Certainly, one of the key factor was that children had more time to experiment in this workshop. Also, having kids facing each other helped in many cases the cross-pollination of ideas - for example when Sophie looked at her brother's machine to get inspiration. Still, the resulting projects were very different from each other, and from the samples.

Looking at Sophie's tinkering process, two facts were particularly interesting: she was seamlessly experimenting in the physical and computational domains, and she experienced the workshop in a very playful and socially engaged way.

Facilitating across physical and computational domains

Sophie experimented across the physical and computational domains, moving back and forth between building and coding, while her project improved and her understanding deepened.

Sometimes, challenges in the physical world became opportunities to play with computation; for example, when she realized that the motor couldn't go any further because the cable was twisted around the structure, she figured out how to program it to make it spin on the opposite direction. Then, when she noticed that the same problem occurred on the other side, she built a sequence of instructions to make it spin back and forth. Other times, exploring the programming blocks led her to discover new possibilities in the physical world, like changing the color of the light.

Even though her experience benefited from the improvement in the design of the activity, probably the most important factor in shaping Sophie’s experience with Art Machines was close and careful facilitation. She was always leading her explorations and deciding what to do - even placing a motor on top of a maker - but my questions as a facilitator helped her to keep the focus, discover new things, and deepen her understanding - through challenges that were always aligned with her zone of proximal development (Vygotsky, 1978). Also, my questions and suggestions were specifically about what she was trying – they felt important and connected enough to what she was doing, that she took them in and integrated the ideas into her machine.
Obviously, such a close interaction between a facilitator and a child was only possible because of the low number of children who participated, the availability of time, and the less constrained nature of the activity. As the number of participants per facilitator increases, it becomes way harder to give adequate attention and support to everybody. Also, facilitation is a fine art, and it is always hard to decide - in real time - when and how to intervene, which questions to ask, and what type of support is more appropriate for the particular situation.

**Personal and social meaningfulness**

Sophie's story also highlights how the emotional and cognitive aspects are tightly interconnected in a tinkering experience. At each iteration, what made her laugh - like the motor getting stuck, twisted around the cable - was also making her think “how can I fix that?” and learn new things.

The Art Machine workshop was an opportunity for her to leverage her inner playfulness. As she was facing the many challenges involved in creating a drawing machine, she never gave up in frustration; on the contrary, she enjoyed being challenged and she dealt with them with a sense of excitement and fun. At each iteration - driven by curiosity and joy - she was testing the boundaries of her understanding and developing a deeper knowledge about physics, motion, and programming.

Also, having her dad and brother around made her experience socially meaningful. She was very proud about her machine: when she saw it moving for the first time, she squealed with delight as she couldn't wait for her dad and her brother to admire her creation.

**Main insights and next step**

In addition to the changes in the activity design, also the less constrained structure of the workshop and a closer facilitation helped creating opportunities for richer learning experiences.

Sophie’s learning story shows that, when enough time is available, and enough support is provided, even young children can engage in playful and very intentional explorations across the digital and physical domains.

This experience inspired me to look more closely - and analyze more deeply - the learning trajectories of children engaged in computational tinkering with Art Machines. This implied relaxing the activity structure even further, toward a free-play experience, and finding a setting in which more time was available.

That’s why I decided to bring a LEGO Art Machines kit with me when I was invited for lunch by Daniele and Radhika: having an entire afternoon to play with their two children seemed to be just the right opportunity for a playful learning investigation.
Case study 3 - Free-play in the living room

Workshop overview

This workshop took place on a Sunday afternoon in a private house. It wasn’t exactly meant to be a workshop, but rather an opportunity to play with Art Machines in an informal, more flexible way, with kids that I knew well, and with the participation of the entire family.

Research goals

This time, the main goal was to investigate how children would engage in computational tinkering with Art Machines, when the experience was presented as a free-play activity, in a familiar context, with no specific time constraints or any expected outcomes.

I also wanted to experiment with facilitation: as more time was available, I wanted to provide less guidance, while still carefully observing the evolution of kids’ understanding as they explored physical and computational materials.

Finally, I wanted to look at the social context of a family, and how people of different ages and different roles - members of the family and friends - would participate and engage with the activity.

Settings and participants

The living room was the place we choose to play with Art Machines. Together with Ray, a 7-year old boy, we set up a big sheet of paper on the floor, where also the LEGO materials box was, along with the WeDo kits and some iPads.

Ray was the first to engage with the activity, and gradually other members of the family joined: Annie, his 9-year-old sister, their parents, and Verena, a 25-year-old family friend.

Ray and Annie had been already programming with Scratch at school, but they had never tried the Scratch iPad prototype before. In a previous conversation, Ray told me that he preferred learning coding with tools like CodeCombat - a text-based coding game. Annie is interested in technology, but she also loves making art and painting with her mom.

Even though I was the only one playing the role of the facilitator, the context allowed for rich social interactions, and many times other adults - while building their Art Machines - helped and got help from kids - asking questions, sharing suggestions, and providing guidance while exploring together.
Observations

Workshop description

The activity started with me and Ray leaving the table right after lunch, as he couldn't wait to check out the Art Machines kit that I brought with me. It ended up with the whole family engaged, after 4 hours of uninterrupted play, and we had to stop only because it was dinner time.

After a short exploration of the materials, Ray was struck by the idea of using 2 motors to create an remote controlled drawing car - as I describe in further detail in the next section. Annie, instead, was fascinated by the possibility to make art, and she experimented with many different designs, tinkering with the structure and with the scripts. Their dad was intrigued by gears and linkages: even though his machine wasn't moving on the floor or drawing, it served to him to explore mechanical concepts, and inspired some of Annie's contraptions. Their mom also joined later: she was carefully observing everyone's work, asking questions and providing encouragement. Verena was playing a hybrid role: as she was experimenting for the first time with LEGO contraptions and programming, she was also supporting other people's work, sometimes with advice, other times providing challenges.

Fig. 4.7 - Ray's family playing with Art Machines in the living room
The diversity of projects created was impressive. The availability of time was crucial in making people create several projects: once an Art Machine was created, and some experiments and drawings were made, people got new ideas and were ready to take it apart and start over.

The context was very relaxed, playful, and rich in interactions. People, especially kids, engaged both with physical as well as with computational materials: Annie, for example, was continuously experimenting with code to make her machines create different drawings.

In the next section, I describe more closely Ray’s learning trajectory over the 4 hours in which we were engaged in playing on the floor together.

Learning story - A (longer) journey with a LEGO car: from physical to computational

Ray’s adventure in computational tinkering started with an exploration of the available materials, gradually moving from the physical to the computational ones, and continued with experiments and investigations, moving back and forth across the two domains.

He was immediately attracted by the LEGO pieces: he took a good amount of time to look at all the different bricks that were available, some of them were new to him and, while he was observing them, he was trying guess how they could be used.

After some time, his attention moved to the WeDo kit, as - guided by my questions - he tried to guess what the different components were supposed to do. While it was harder to figure out the function of the sensors, he soon recognized and got excited about the motors, and started looking for some wheels. As he realized that they were not included in the kit, he decided to use two gears as wheels, just as the two girls had done in the first workshop I facilitated. This time, though, there was much more time to experiment. He excitedly plugged the motors to the hub, expecting them to move, but he was disappointed to see that nothing happened. It was time to explore the computational material.

I encouraged him to check out the Scratch app, and he promptly recognized the motor instruction blocks. In the beginning he thought that the number below each block - that indicates for how many seconds the motor should spin - was indicating which of the two motors to control, so he used two blocks, and put 1 and 2 as parameters. To his disappointment, only one motor started spinning, and I had to confess to him that the iPad app only allowed to control one motor. It wasn’t a big deal, and he continued exploring the computational capabilities of the toolkit.

In a continuous process of guessing, trying, observing, and adjusting his ideas, he discovered many of the capabilities of the programming language. He discovered the random light color block, and was amazed to see the light color changing when he tried it in a forever loop. He
correctly guessed the function of the "when-motion-detected" block, and he decided to try it out, activating the light-changing loop when he waved his hand.

When he got more familiar with the programming language, I encouraged him to put the machine on the floor, to check if it could move. At that point, the motor was electrically plugged but not mechanically connected to the hub. Building a structure to keep the motor stable in place, was particularly challenging. Ray tried numerous strategies, until he had an intuition: he connected it laterally, and he added another motor - "even if it doesn't work" - just to make the machine more stable and symmetric. Still, the machine didn't move, as the wheels were not touching the ground, so he cleverly flipped the entire machine upside down. As one of the motors started running, the machine started spinning in circles, to everyone's excitement. At this point, it was easy to add a marker, and beautiful perfect circles appeared on the paper floor.

At this point, I brought up the fact that he could control the 2 motors using the standard Scratch version on my laptop, and he was happy to switch to it, as he was already familiar with Scratch. I helped him setting up the connection and he started exploring the Scratch extension blocks. In few minutes he made what software developers call a "porting": he translated the script he had on the iPad - that uses a reduced set of instructions and a horizontal iconic grammar - into a regular Scratch script.

Fig. 4.8 - Ray debugging the Scratch code for his Art Machine
From this point on, he didn’t change the mechanics of the Art Machine, and he focused on his code, for example creating two different procedures to control the motors and the lights. Then, he realized he could use the Scratch stage to give instructions to the user: when the program started, a Scratch cat on the screen was instructing the user on how to connect the WeDo hub and make the machine move.

I left Ray continuing his work with Scratch by himself, to have a chance to look also at Annie’s amazing Art Machines. When I came back to him, he was working on one of his old Scratch projects. The LEGO Art Machine activity had been a way for him to rediscover and reconnect to Scratch.

I was very happy to see the Scratch projects he shared with me, and soon we found ourselves creating another project together: this time we wanted to make his Art Machine remote controllable using the keyboard arrow keys. When we tried it for the first time, something was odd, it wasn’t turning as we expected. I left to him the hard fun of debugging: in few minutes, he found out what was wrong, corrected the bug, and enthusiastically drove the LEGO drawing car around on the paper canvas.

Now that the machine could go straight, turn, and make circles, Verena challenged him: “Can you make it draw a circle, then a line, another circle, and come back... automatically?”. “Sure!” was his answer, as he jumped into creating the script, that involved a complex coordination of instructions involving motor directions and timing.

Unfortunately, our time was over. We had played for almost 4 hours, and it still took some negotiations - and the perspective of a yummy dinner - to convince everybody to get up from the floor: another unmistakable indication of how engaging it was for the children - and some adults as well - to tinker with Art Machines.

Analysis and reflections

The main takeaway of this session was to realize how crucial it is for people to take their time to explore and experiment, and how the available timeframe impacts the scope and duration of tinkering iterations. Also, it was interesting to look at the social aspect of the activity, as it brought the family together, favoring a rich context for interactions.

Make time for exploration, and room for experimentation

In the first workshop I facilitated at the girls’ coding club, we only had less than 1 hour to introduce the activity, explore the materials, build something, and share it with others. The timeframe was extremely constrained, and kids didn’t have enough time to get to know the physical materials and the computational capabilities of the toolkit. Also, one of the goals was to
have every group make an Art Machine to share at the end of the activity; that's why, as a facilitator, I often had to push on quicker iterations and faster experiments.

In this activity, instead, people were able to work at a natural pace. There was no pressure about reaching a specific outcome, and they had enough time and space to explore possibilities, to take more risks, and to learn from mistakes. Having enough time to explore is essential in any creative learning activity, but it looks particularly important when the available materials belong to two different domains, the physical and the computational, in which affordances are inherently different.

In addition to having more time, the less structured nature of the activity allowed people to explore more widely the space of possibilities, and to do experiments more freely. Of course, the activity still had an inherent structure, and the goal was to make an Art Machine. But by adequately relaxing constraints and expectations, Ray and other family members were able to find their personal pathways into the activity, as it is shown by the variety of project that they were able to make.

Finally, having more time allowed me to understand how engaging the activity can be: even after 4 hours of uninterrupted play, it looked like everyone would have been happy to keep going for many hours.

**Iterations in physical and computational space**

The extended timeframe and less rigid structure of activity also helped me to get a better sense of how a natural flow between physical and computational explorations can look like, where people are not rushed to build something with materials they are not familiar with. Ray dedicated almost one hour to explore the LEGO materials and to try out the available instructions on the iPad, before putting his first prototype on the floor. And it took another hour of trials and errors to find a good solution for the structure of his Art Machine, and another one to make experiments with code.

It's also interesting to notice that Ray - and every other participant - started building with the physical materials, and then got interested in the programming possibilities. And usually this sparked new explorations in the physical domain, and then new experiments with code. Each of these phases takes a considerable amount of time, and it can be considered a domain specific macro-iteration. These macro-iterations, often include some back and forth between physical and digital, but it easy to identify that, at least in the beginning, personal explorations take place dominantly in one or the other domain.

Also, it appears that the duration of each iteration tends to decrease as people get more familiar with the materials. Once the affordances and capabilities in each domain are clearer, it's easier to jump back and forth between physical and digital, and iterations become faster and more frequent.
A Samba School in the living room

The social environment and the type of interactions that emerged in the living room, reminded me of what Seymour Papert called a Samba School (Papert, 1980). People of mixed ages, with different roles, and with different familiarity with the tools, were all engaged in a common activity.

In this environment, ideas circulated in an organic way, and group collaboration happened naturally. At one point, also my role shifted from a facilitator to a peer, for example when I worked together with Ray at the Scratch code that he helped me debug. The result was an engaging, rich social experience that everyone enjoyed.

Main insights and next steps

Relaxing the constraints of the activity was extremely informative, in particular because it helped me to:

- re-adjust my expectations about what is possible to achieve in a limited timeframe;
- gain new insights on the dynamics of computational tinkering iterations;
- focus on the quality of the social interactions in computational tinkering experience.

The next question for me was: "How can I apply these insights into a more structured and time-limited experience?". That's why I decided to get back to an after-school program, proposing the activity in the context of a Scratch after-school coding class, hosted at the Harvard Ed Portal.

In addition to applying the lessons learned so far about the design of the activity, I also wanted to create more opportunities for students' self-reflection, and experiment with different ways to document the experience.

Case study 4 - Workshop at Scratch coding club

Workshop overview

Research goals

In addition to applying some of the insights gained in the previous activities, particularly in terms of setting expectations in line with the available time, in this activity I also wanted to engage kids in self-reflection and try to understand how they saw themselves, in relation to tinkering and computation.
Therefore, I designed two interviews: one was in the form of a pre-session activity where kids expressed their positions situating themselves in different places of the room; the other was in the form of a discussion at the end of the session, reflecting back on their work.

As the interview provided very interesting insights, in the rest of this section, I will focus more on that, rather than describing the Art Machine activity.

**Settings and participants**

Harvard Ed Portal is a community learning center that offers a variety of programs for youth and adults. Among their programs, they host a computer coding club for students between 11 and 14 years old, once a week, where they learn how to create games and animations with Scratch.

As a part of this program, I proposed to have kids explore the possibility of connecting Scratch to the physical world, creating and programming LEGO Art Machines. So, instead of using the iPad interface, kids used the standard Scratch interface, through the LEGO WeDo extension.

Seven kids participated to the activity: they all had some familiarity with Scratch, as they had all attended 5 previous sessions in the code club. In addition to me, another facilitator and 2 more adults were available for support. The session lasted 1.5 hours and included semi-structured interviews, before and after the activity.

In order to integrate and support my observations of children's activity, I recorded audio and videos of the entire session. Capturing their words and actions helped me to focus on the actual facilitation and provided me with documentation I could analyze at a later time.

**Observations**

**Pre-workshop interview**

With this activity, I wanted to investigate how kids saw themselves along different dimensions, including their tinkering vs planning disposition, their preference about working with physical or computational materials, and their inclinations toward following instructions or figuring things out. Instead of a formal, one-on-one interview, I proposed an activity in which kids were answering the questions placing themselves along a line between two opposite statements: for example, "I like to create with computers" vs "I like to create with physical materials". After kids positioned themselves, I asked some of them to explain why they choose that particular spot in the spectrum.

This initial activity was meant to provide a context for reflection: as they were answering and listening to other people's answers, they were reflecting on their own way to engage in physical
and computational creation. Also, some of the answers they gave initially were proposed again at the end of the session, and provided an additional way to reflect back on the Art Machine activity they had participated in and provided a way for the kids to compare their initial thoughts with their actual behavior/mindset during the tinkering session.

Here, I will focus on three of these dimensions, reporting some of the answers kids gave and comparing them with their behavior during the activity.

**Tinkering vs Planning**

One of the questions was: “When I build things, I know exactly how they will look at the end / I like to try different things and see what happens”.

Kids spread over the entire spectrum. Lisa and Mark were the most extreme tinkerers, while Victor positioned himself as a planner.

Lisa: “It’s fun when you play with the materials and put them together, you discover things and you have more variety of things you can make. If you choose something to make you don’t really have a variety of things that you are thinking about, you just have that thing that you want to make.”

Mark summarized his similar position as: “If I know exactly what it’s going to happen, it’s not going to be fun”.

On the other side Victor saw himself as a planner and related this to the fact of being very precise: “I’m a very precise person, my friends make fun of me! When I’ve something in mind, my project will resemble what my original idea was. There will be things that I change, but the general idea will be the same as how I imagined it.”

Among the kids who positioned themselves in the middle, they did it for different reasons. For example, Barbara stated that she usually sticks to her plan, even though she might change something here and there. Eliza, instead, saw tinkering as a first, explorative phase, that helps her coming up with a plan: “I always start trying out different things, and then I get to the point in which I know exactly what I want, kind of exactly what I want”.

**Following instructions vs figuring things out**

Another question was: “When I assemble a kit: I like to follow instructions / I like to figure it out without instructions”.

This time, Lisa and Victor were on the same side: they both declared they prefer to figure things out by themselves, and referring to instructions only when they are stuck. Victor said: “I don’t
want to read the instructions, I think that instructions can be boring, it’s much more fun to figure it out by yourself; but if I can’t figure it out by myself, I would use the instructions.”

On the opposite side, Mark, despite preferring not having a plan, declared he liked to start by following the instructions: “To get to see if it’s going to work or not, first you have to follow the instructions, otherwise you don’t know if it’s going to work very well, [...] and when it’s done you can add more things”.

Building in the physical / building with computers

Another question was about their preference about building with physical materials, like LEGO bricks, or building with computers, for example using Scratch. In this case, kids' positions were generally less extreme.

Victor was the most enthusiastic about building with computers: he mentioned that he got his passion from his father, who is a software developer. Also, he liked Scratch because of the availability of blocks: “With Scratch everything is there for you, you can use everything for free, but with legos you need to buy that small part that will make your building better. With Scratch you don’t need to buy anything.”

Lisa, added that with Scratch and technology you can “control and make things move [...] while with Legos it’s more like a structure”.

On the other side, Eliza pointed out that she liked both but: “With Scratch sometimes you have the block but you don’t know how to use it”.

Workshop (short) description

After the initial interview, I gave a quick demo about how to use the Scratch extension blocks to program the motor and interact with sensors, and I invited kids to explore the rest of the Scratch blocks and the materials on the table before starting working on their Art Machines.

Kids divided into small groups, and started working on their machines. As expected, due to the limited time available, most of the explorations involved the physical aspects of the artifacts, and the code they created was generally very simple. Even though kids had the opportunity to use the Scratch stage to create interactive animations, or use the computer keyboard to interact with the machine, their attention was generally drawn to the physical domain and they didn’t consider to combine their physical creations with more elaborate Scratch projects.

Post-workshop discussion

The final conversation was an opportunity for everyone to reflect back on their learning process.
Kids were commenting on each other’s artifacts, and also sharing some of the most challenging parts of their explorations.

Despite the answers given in the initial interview, all the kids agreed that building their Art Machine with physical materials was what they enjoyed most.

In particular, Victor, who had previously stated his preference on creating with computers, explained that: “There were only 10 different blocks to use. […] I think it is more fun to make a [physical] design”.

His answer is not surprising, as in all previous workshops kids got very engaged in building the physical structure of their Art Machines, especially in the beginning. At the same time, it suggests that he perceived the Scratch extension palette as too limited, and he didn’t consider the possibility of having an animation on screen as a part of the project. That made me realize that a different modeling of the initial samples - for example using the Scratch stage as a part of the sample project - could have brought to wider explorations and more diverse outcomes.

Analysis and reflections

Even though the sample was not statistically significant, children’s answers provided great insights on how they see themselves as makers, and help to realize the complexity of the relationship between tinkering and planning, and between physical and computational materials.

A surprising discovery was made regarding the relationship between tinkering and following instructions. My initial assumption was that tinkerers would prefer to experiment and figure things out by themselves, while planners would be more inclined to follow instructions. Instead, even in this small sample, I’ve found some tinkerers who like to follow instructions, and some planners who prefer to figure things out. And all of them provided convincing motivations to explain their preferences.

Another interesting fact is that kids introduced the dimension of time to explain how their attitude might change over the course of a project. Most of them didn’t identify themselves as “pure” planners or tinkerers, but they described how they can use one or another strategy / mindset according to the phase in which they are with the project. For example, some of them start with tinkering until they choose a project to stick on. Others start with a plan and change their mind as they interact with materials or get new ideas.

A last reflection is about physical and computational material affordances. Kids recognized that computational blocks have the ability to build behaviors, not only structures. This certainly represents a big advantage over the physical blocks, but it also brings new challenges, as it is not always easy to understand what is the effect of an instruction, and figuring it out is less natural or obvious than doing it in the physical world.
Computational tinkering with LEGO Programmable Art Machines

Pseudo random scribblers and predictable turtles

Scribbling machines are inherently unpredictable: a small change in their structure often results in a big change in their motion and, consequently, in the drawings they generate. This happens because, despite the simple material they are made with, scribbling machines are complex machines. Their behavior results from a combination of a huge number of physical variables - distribution of weight, position of the motor, grip on the floor, and so on.

Computers, instead, are generally perceived as very predictable. They just execute the instructions they receive, so the effect of a computer program is expected to be unsurprising. Of course, the effects of a program are sometimes unpredictable without any interactions with the physical world. But with LEGO Programmable Art Machines, interactions with the physical world make things even more unpredictable. Therefore, programming an Art Machine often consists of a trial-and-error exploration of parameters like the motor speed and direction. That makes Art Machines a context in which people can tinker with computation.

On the other side, as it emerged in some case studies, programmability also evokes and enables a new category of more deterministic Art Machines. Some of the children - like Emma, Jenny, and Ray - recognized that they could use the toolkit to make a fully controllable steering vehicle. What they called "cars" were nothing else than a modern, kid-made versions of Seymour Papert's LOGO-programmed turtles. These turtle-like Art Machines can be a context in which people explore - and intentionally apply - computational ideas into a physical tinkering activity.

Physical first (and foremost)

In all the sessions I facilitated, children immediately started by building with physical materials, and only explored computation in a later moment, usually when they were completely satisfied with the design of their machines.

Obviously, kids had to build something before they could start programming it. Especially when the iPad interface has been used - where no stage is available for screen animations - programming only made sense when there was a physical object to program. This might explain why kids started with physical explorations, but it doesn’t help with understanding why they got more engaged with physical building. Even children who generally prefer to build with code, like Victor, spent most of their time playing with physical materials and admitted that they enjoyed the physical construction more than programming.
Many factors might have contributed. Some have to do with the different affordances of physical and computational materials: we’ve manipulated physical objects since we were born, while understanding what programming instructions do - especially when the programming interface is an early prototype - can be trickier. Making the programming interface more intuitive - for example curating the instructions available according to the specific activity - and giving kids more time and guidance on the computational tools, would certainly help. Another factor is related to motivation: even when children (or adults) completely understand what is possible to make with computation, we need to provide them engaging reasons to use it. To cite an expression often used by Andrew Sliwinski, in this phase we were able to provide opportunities to go “from zero to hello world” with computation; the next step will be to fill the gap between “hello world” and “wow”.

From controlling to programming

In most of the cases, children’s first experiments with the programming interface started with very simple interactions: they just used one instruction to turn the motor on and off, tapping on the corresponding block. Even kids who explored more of the available instructions often used the iPad interface more as a control panel, rather than a programming environment. This ability to remotely control their physical artifact seemed to be engaging enough for a first computational exploration, and most of the kids concentrated more on building and controlling their machines, rather than programming them.

Kids who experimented more with programming also encountered obstacles. Some of them created very elaborate scripts, including long sequences of instructions and nested loops, but they couldn’t explain what their scripts were supposed to do. While snapping together instruction blocks was easy, understanding their effect was less obvious, and it often required a good amount of time and facilitation.

When adequately supported, though, kids understood and leveraged the fact that computationally-controlled variables could be programmed to change over time. This characteristic enabled them to create more elaborate, yet not completely predictable, drawings. One example is the “palm tree” drawing machine that two girls made in the first workshop, using a script that periodically changed the motor direction every few seconds. In the case of turtle-like machines, programmability opened up the possibility of making more intentional drawings, like the combination of circles and straight lines that Ray was programming for his LEGO car.

Finally, some children also used sensors to trigger behaviors. The programming interface we used for LEGO WeDo still has some limitations; for example, the distance sensor only detects whether an object gets close, and the tilt sensor only determines the direction of the inclination, without providing numerical values. Nonetheless, sensors open up new possibilities for interaction, introducing an element of magic and wonder that can definitely lead to “wow”.

64
5. Looking back, looking forward

In this chapter, I reflect on my explorations. I look back to identify some key characteristics of a computational tinkering experience, looking at how children learned in the activities I designed and facilitated. I look forward to propose directions for further investigations and development.

Reflecting on Learning through Computational Tinkering

A computational tinkering activity is one in which the learner is actively engaged in creating through tinkering, building with both physical and computational materials. Observing children engaged in the activities I designed and facilitated helped me to identify some of the key characteristics of what learning looks like in computational tinkering experiences.

Learning by iterating

One of the key characteristics of a computational tinkering activity is the iterative and incremental process of making in the physical and computational world - which corresponds to an iterative and incremental process of making sense in the learner’s head. This process is wonderfully exemplified by the Creative Learning Spiral (Resnick, 2007), in which the learner is continuously engaged in creating something, experimenting with the artifact, getting new knowledge by interacting with it, sharing, and imagining new possibilities for the next cycle - to fix something, improve the design, or steer the exploration in new directions.

When both physical and computational materials are available, iterations happen across the two domains in different ways. For example, Ray started by iterating multiple times on the physical design of his LEGO car, before he was satisfied enough to start experimenting - and iterating - with code. Sophie’s process, instead, involved a continuous back and forth between the digital and physical: every time she tried her Art Machine, she decided whether she wanted to change something in the structure, or something in the code. Sometimes, especially after participants have developed familiarity with tools and materials, iterations seamlessly span across domains: while working on a physical structure, kids change something in the code; while working on a script, they adjust something on the physical artifact.

Regardless of the specific way in which learners iterate - or the scope and duration of each iteration - a computational tinkering activity always involves trying things out and debugging, in
the computational as well as in the physical domain. It is in this bottom-up process of constantly trying something, stepping back, noticing what is happening, and trying something new, that learning happens.

Learning by exploring and experimenting

In a computational tinkering activity, learners are continuously engaged in exploring and experimenting with materials and ideas. In the Art Machines activity, for example, it was natural for children to get started by examining the available materials - i.e. the LEGO bricks and the instruction blocks available in the Scratch programming interface - in order to understand affordances and possibilities, and to get inspirations for their projects.

Exploring materials involves experimenting with them. For example, Ray engaged in a very systematic review of all the available Scratch instructions, trying them out one by one; other kids chose only a subset of the instructions to experiment with. Experimenting with instructions, in turn, helped children to generate new ideas about how to use them in their projects, leading to new explorations and new experiments.

This process of exploring and experimenting is crucial, because it is the way in which people develop their understanding of tools, materials, and concepts. For example, Sophie understood the computational concept of sequentiality not by listening to an explanation, but by experimenting with programming blocks to make her motor spin back and forth. When she realized that instructions were executed in order, one after the other, she was able to use the same concept in many other ways.

Papert (1996) calls it “the power principle”. He asks: “what comes first, using it or ‘getting it’? The natural mode of acquiring most knowledge is through use leading to progressively deepening understanding. Only in school [...] is this order systematically inverted. The power principle re-inverts the inversion.” (p.98)

In computational tinkering, ‘using’ computational ideas often comes before ‘getting’ them: exploring and experimenting are the fundamental practices that lead to discovery and, in turn, to understanding.

Learning by reflecting

Computational tinkering involves learning with computation by iterating through explorations and experiments. But what distinguishes computational tinkering experiments from a random process of trial-and-error - or computational “just-tinkering” - is the reflection that happens at every iteration, when the learner steps back, notices something - expected or not - and then
intentionally decides what to do next. To put it in Ackermann's words (1996), it is "the ongoing
dance between diving-in and stepping-out" (p.28).

While observing children building Art Machines, I've often found myself wondering what was
going on in their minds, and how intentional were their explorations. Most of the times I asked
"What are you trying to do right now?", the answer was "I don't know". But, as I watched more
carefully, I realized that their actions were often guided by intuitions that they weren't fully aware
of, that were hard to verbalize or that they didn't feel ready to share.

With computational materials, the challenge of stimulating and recognizing intentionality was
even harder. Some of the scripts that children created, especially during their initial explorations
of the computational space, appeared to be a random collection of instructions. For example,
Emma created a very complex script to program her Art Machine, with several instructions and
nested loops. To my question: "What does the script do?", she simply answered: "It makes the
motor run". Going through the single instructions, and trying to verbalize the expected behavior
together, gave her a better sense of what her script was doing, and helped her debugging and
planning her next experiments more intentionally.

Creating opportunities for reflection and making intentionality visible were big facilitation
challenges. Asking specific questions at the appropriate time, observing carefully participants’
actions, helping them verbalize what they did try and why, having a structured discussion at the
end of the activity - all these strategies proved to be helpful, but it is important to notice that
there is no silver bullet. Facilitators of computational tinkering activities should always be on
lookout for new ways to elicit reflection, iterating and experimenting with facilitation strategies.

Learning through a personal, social, and emotional journey

In all the workshops I facilitated, participants experienced the activity in their own personal way.
Personal doesn't only refer to personalizing their Art Machine with pipe-cleaners and googly
eyes. The specific materials they chose to work with, the goals they set for themselves, the
challenges they faced, the ideas they encountered and applied along the way - they all
contributed in defining a unique learning journey for each of them. These trajectories were rarely
straight lines. Sometimes kids got to the same concept through different pathways; other times,
their trajectories simply led them to encounter and learn different things.

In their personal explorations, children learned from each other, too. In the context of a
workshop, their learning trajectories intersected in different ways: working together on a project,
looking at other people's work for inspiration, asking each other questions on how to do a
certain thing.

The metaphor of a personal journey is not only limited to the cognitive dimension: it also refers
to the journey across feelings and emotions that the learner encounters along the way. For
example, while trying to make her Art Machine move, Sophie went through moments of frustration and disappointment when things didn’t work as expected, but also immense joy and contentment when she reached her goals or discovered something new. Also, she was sharing her delight with people around her: her dad, her brother, and the other kids at the table.

As researchers at the Tinkering Studio show (Petrich et al., 2013), emotions like frustration, joy, disappointment, or pride are indicators that kids are truly engaged in the tinkering activity. This engagement is not just a nice-to-have byproduct, it is an essential part of how people learn through tinkering. The name “computational” shouldn’t be misleading: for computational tinkering - as for Constructionism - “learning is more related to love, than logic” (Papert, 1983).

**What is special about computational tinkering?**

All the considerations I shared above - about learning by iterating, exploring, experimenting, reflecting, in the context of a personal, social, and emotional journey - are not new concepts. Those ideas are at the core of any constructionist, creative learning, or tinkering experience. The reason why it is important to highlight them is to emphasize that those remain the essential elements of a learning experience, and that we shouldn’t lose or forget them when computation comes at play.

In computational tinkering, computation is one of the materials people can tinker with. But not all materials are created equal, in the same way not all tools have the same potential. As Papert notices in the foreword of “Turtles, Termites and Traffic Jams” (Resnick, 1997, p. x), it is important “to distinguish between tools (reasonably described as “just tools”) that improve their users’ ability to do pre-existing jobs, and another kind of “tool” that are more than “just tools” because of their role in the creation of a job nobody thought to do, or nobody could have done, before”.

Computational material has very unique affordances, and it has the power to add new, original dimensions to the tinkering experience. For example, computation expands the possibilities for tinkering with processes that play out over time - i.e. to dynamically change the motor speed and direction in a programmable Art Machine, or to create a sequence of colors in a Digital Lightplay installation. Computation introduces the ability to make artifacts interactive through the use of sensors, or to engage with mathematical concepts like randomness. Ultimately, computational material is a unique medium for learners to reflect about their own mental processes, as the code they write is a representation of their own thinking.

One of the challenges of designing computational tinkering activities consists in creating the conditions to leverage these unique computational affordances, allowing learners to engage with new ways of making and thinking.
Future directions

In this last section, I identify two major directions for future work. One is related to enhancing the role of computational materials in the activities, identifying ways to leverage computation to expand the space of creative possibilities. Another is related to expanding the reach of computational tinkering experiences beyond informal learning settings, overcoming the challenges of more traditional learning environments like schools.

Making computation more accessible

As I've illustrated in analyzing the case studies, kids engaged with the activities focusing more on the physical side of building than on the computational one. However, I believe that computation has much more to offer in this kind of learning experiences. Therefore, a first question to be further explored is: “How can we make computation more accessible, engaging, and ultimately more meaningful, in a computational tinkering experience?” To start dealing with this question I’m going to use a metaphor broadly used in the design of construction kits for kids, proposing ways to lower the floor and widen the walls (Resnick & Silverman, 2005).

Lowering the floor - i.e. making the programming interface more intuitive, and providing adequate scaffolding through learning resources and facilitation - would undoubtedly help. But in addition to lowering the floor, it is necessary to carefully select the tiles the floor is made with. In the same way we curated the physical materials - for example, choosing the LEGO bricks that are included in the Art Machines kit - it would be helpful to curate the palette of instruction blocks available for each activity. In my first experiments, I've been using a generic programming interface for the LEGO WeDo, but having a tailored set of instructions - in the spirit of Papert’s Microworlds (Papert, 1987) - would probably make it easier for kids to engage with computation, removing some of the obstacles related to understanding how the system works and how it can be programmed.

Also, widening the walls - expanding the different ways in which computation can enrich the activity - could provide more opportunities to get to the “wow” effect. One promising direction for Art Machines, but also for Sound Machines and Lightplay, would be to leverage the computer screen to create projects that have both a physical and digital dimension. Also, sensors can provide new ways to interact, and my intuition is that with the Programmable Art Machine we have only scratched the surface of the possibilities: with new toolkits becoming available - like the LEGO Boost or new Scratch-compatible devices - there will be even more exciting possibilities to leverage computation in the physical world.

In addition to enhancing the tools, more can be done to improve the structure of the activities, taking into account that computation can sometimes appear obscure or intimidating in the beginning, and that exploring the computational domain often requires a significant amount of
time. For example, activities could start by inviting children to experiment with pre-built artifacts, to help them familiarize with the possibilities that computation offers, and only at a later time encouraging them to change something or make something new in the physical world. More generally, the design of the activity should take into consideration that computational materials are normally less intuitive than physical ones, and the process of understanding them needs more time and adequate scaffolding.

Bringing computational tinkering in schools

The learning contexts in which I experimented with computational tinkering activities were informal settings, mostly after-school programs, with a very limited number of children and a good amount of help in terms of resources and facilitation. I chose those environments because I wanted to have an opportunity to look closely at how kids learn in a computational tinkering activity, and iterate quickly on the design of an introductory activity. The long-term goal, though, is to make this approach broadly available in more traditional settings, namely, in schools. To do so, it’s crucial to identify the challenges and investigate ways to help teachers bringing computational tinkering in their classrooms.

One of the first questions I usually get from teachers and educators who want to bring LEGO Programmable Art Machines or Lightplay to their students is about the cost of the hardware. Unfortunately, many schools around the world can’t afford to buy expensive devices or access MIT prototypes. But the cost of hardware is rapidly decreasing: it’s already possible to develop activities and toolkits that leverage the low-cost, accessible hardware already available on the market. For example, using Arduino, sensors and servo-motors, it is possible to achieve similar functionalities of a LEGO WeDo kit at a fraction of the cost. Developing low-cost toolkits and activities around them is a crucial next step for bringing computational tinkering to a broader audience. On the other side, it is important to remember that computational tinkering doesn’t necessarily involve using external devices: a programming environment like Scratch has all we need to start experimenting and tinkering with code.

Even important than developing low-cost technology is supporting motivated teachers. Despite the resistance that certainly still exists toward less traditional approaches in education, an increasing number of teachers and educators see the value of learning by tinkering and embrace constructionist ideas, especially in using technology as a medium for expression. How can we enable and support them in becoming designers and facilitators of computational tinkering activities? In addition to continuing to spread the underlying constructionist ideas, it is vital to provide teachers and educators with actionable resources, facilitation strategies, activity guidelines and examples that they can remix and adapt to their classrooms. Also, more work needs to be done to support the creation of local and online communities in which educators can exchange ideas and share their experiences among peers. A good model to look at is the
ScratchEd online community (http://scratched.gse.harvard.edu/), where educators who use Scratch can share stories, exchange resources, and connect with other people.

As the interest in teaching computational thinking skills to young people continues to grow in schools, it is essential to prevent coding and computation from getting “school-ified” into just another subject. Computational tinkering represents an opportunity to shift the perspective on the creative opportunities of building with code, and to focus on tinkering as a universal learning approach, based on a playful engagement with any type of materials and ideas.

A final note

The idea of computational tinkering is still in its infancy, and filling up this term with meaning is a collaborative, iterative, exploratory, and playful effort that many people are making together.

In the same way, my explorations and the process of writing this thesis has followed the same approach, with goals continuously shifting and the structure evolving as I was getting new ideas by observing children, talking to people, reflecting on my writing.

I consider this thesis the result of my first macro-iteration, a not-finished-yet project that has been a helpful object-to-think-with. My hope is that this object, and the ones that will come, can contribute to the broader conversation on computational tinkering, bringing a perspective that can inspire others to design activities and technologies for children in the same playful spirit.
Acknowledgements

A huge list of special people provided inspiration, guidance, feedback and support in many different ways during this hard-super-fun journey of being a student in the Lifelong Kindergarten.

I have no words to fully express my admiration and gratitude to Mitch Resnick and to Natalie Rusk. Thanks for your time, for your support, and for the thoughtful conversations that contributed to this work and to my personal growth. Thanks for being incredibly inspiring role models, caring mentors, and loving human beings. Thanks for believing in me and in “kindred spirits”: being adopted in the LLK family has been the greatest privilege of my life.

A big thank you to my readers, for their inspiration, feedback and support. To Karen Brennan - thanks for sharing your contagious enthusiasm for learning by creating, and for being the living demonstration that teaching - actually, designing for learning - is an act of love. To Karen Wilkinson - thanks for introducing me to the art of tinkering, for your thought provoking questions and for your wholehearted support. To Sep Kamvar - thanks for your encouragement to challenge assumptions and to reshape the world, driven by values and purpose.

This work is part of a larger effort of supporting Tinkering in the Digital Age: learning with the Tinkering Studio, Reggio Children, and LEGO Foundation is for me an extraordinary opportunity.

Amazing students from HGSE joined me as interns: Lily, Karishma, Michael, and Clara - thanks for your help, for your insights, and for enthusiastically embracing our continuous changes of direction! And thanks to all the young people - of all ages - who tinkered and played with us.

The Lifelong Kindergarten group and the Scratch Team have been my family here. Special thanks to Andrew, Eric, Jie, Jennifer, Shruti, Stefania, and Kreg - it's great to learn from you, and with you. And to past students Ricarose, Sayamindu, Champika, Alisha and Juliana - your work has been a great place to start tinkering with this thesis. A super-awesome special thanks-back to my partner in play - Moran. You know I couldn't have done it without you. Literally!

Thanks to the people who make the Media Lab community so open and creative. And thanks to all my friends here, especially to Shruti for her warmth, to Laya for her thoughtfulness, to Manisha for her enthusiasm. Thank you Mina, for being deeply close.

Thanks to many cool people in Italy, for what we've done, and for what we'll do together. Special thanks to Angela and mentors in Bologna for keeping up the coolness. To my friends Peppe and Daniele for keeping in touch, to Beatrice for pushing me forward. To my elementary school teacher Anna Maria, for seeding in me the love for learning, and the joy of helping others.

A final thanks to my beloved family: mamma, papà, brothers, cousins, aunts, uncles, and my dear nonnicchia, thanks for your unconditional love. I always carry it with me, wherever I go.
References


