Programming For The Natives: What is it? What's In It For The Kids?[1]

Edith K. Ackermann
School of Architecture. Massachusetts Institute of Technology, Cambridge, MA., US
http://www.media.mit.edu/~edith

Abstract

Programming is many things to many people, and not everyone agrees on its potential for human learning. This is especially true at a time when even very young children are increasingly "expert" gamers, tweeters, information-seekers, and digital "bricoleurs". Mostly self-taught, or at least grabbing much of what they know outside the classroom, today's youngsters (also referred to as "digital natives") indeed surprise—and on occasion surpass us—with their clever uses of all things digital. Question is: how much of this "expertise" is deemed sufficient by experts in the field? This paper looks at programming as an opportunity to address issues of agency, control, and interaction styles, as played out in the creative and critical uses of "smart" tools by curious minds. The focus is on views and uses of "programming" as a means for 1) making things do things *(instruct them to follow and execute orders); for 2) "*animating" things (*endow them with a mind of their own, teach them to "look out for themselves"*); *and for 3) "*poking" things *(modulate how things act or interact by tweaking some parameters in their environment). I present settings where youngsters are invited to give and execute orders, to take over or relinquish control, and to compose with things endowed with a "logic" or integrity of their own. I draw lessons for the design of programmable play kits for young children.*

Keywords

Program, model, simulate, animate, object-relation, agency, control, bricolage, children.

Introduction

At a time when computational devices have become an integral part of our lives, and make it easier to run programs, model interactions, and simulate behaviours, people's ideas of what programming, modelling, or "simuling" are about are deeply changing, as are their ways of relating to existing authoring and editing tools. More than in the past, performance and simulation are granted a new place besides language, and there is no doubt in most people's minds that downplaying the significance of *new media literacy* (literacy *beyond print*) would be today's equivalent of promoting illiteracy. What is less

[1] An early version of this paper was presented at the "Constructionism 2012" Conference, in Athens, The current version has been further polished. Photographs have been added. It is available at CRN. URL: http://www.childresearch.net/papers/digital/

clear, to this day, is the status of programming itself and its alleged benefits in helping youngsters acquire the competences they need to become fluent ITC users. In other words, how deep under the hood should we be looking in order to meet so-called 21st century skills requirements? What's there to be gained in the first place? And what's in it for the children?

The purpose of this paper is to shed light on the intricacies between modelling, simuling, and programming, from an experiential perspective. Building on research with, and observations of, children, we focus on issues of agency, control, and interaction styles, as manifested in the creative and critical uses of "smart" tools by curious minds. We look at how children, and adult-experts, use and think of "programming" as a means for exploring and optimizing the interplay between people and *actionable things*: in this case, any artefact, physical or digital, that is either seen as being responsive or inner-driven, treated as if it were, or made to stand on—or do things— on its own.

I discuss why exploring the "logic" of gives and takes, through interacting with, relying on, or controlling "smart" toys can enrich our experience and understanding of "programming" (in the broad sense of "making things do things"). I present settings in which youngsters are asked to use objects as models, give and execute orders, take over control and let go of it, animate things, and simulate behaviours. I draw lessons for the design and evaluation of "programmable" play [-learning] kits for young children.

New media ecologies, new genres of engagement: the changing relations between today's youngsters and their artefacts

Animated toys have always occupied a special place in people's lives. They are intriguing because they do things. Sometimes they even seem to have a mind of their own. Many are responsive to our solicitations. In all cases, objects that behave are treated differently than inert toys. Obviously, toys need not be animated to *be "made to behave"* in a child's imagination. In their pretense play, children endow things with life all the time. Puppets, dolls, stuffed animals, and even sticks and brooms are made into living beings, and endowed with all kinds of special powers. Yet, toys that *actually* behave, I posit, elicit new ways of relating, and are used in different ways (Ackermann, 2005, Turkle, 1984).

Fig.1a-*Aibo and I. Photo Florent Aziomanoff*

Fig 1b - *DuckyDo follows me. Photo by author*

Things that do things, and "telepathic" toys

T*hings that do things* are objects-to-think-with, in Papert's sense, yet of a particular kind (Papert, 1980). They intrigue us because of 1) their hybrid nature (look like things yet act like people); 2) their relative autonomy (responsive but with a mind of their own); and 3) their singular form of "smarts" (different but intriguing and sometimes even "loveable"). It is their very artificiality and thus "forgiving" nature that make it possible to explore, enact, and work through issues of identity and object-relations, and to learn about how different creatures [people, animals, plants] or things [stones, tools, machines] act in the world, communicate amongst themselves, and respond to a child's solicitations.

What I call *"telepathic" toys* have this additional property that they respond to our solicitations at a distance. For example, if I push a button on my controls, a cartoon sets itself in motion on a TV screen, at the other end of the room; and as I zap between channels, I can make things appear and respond remotely. No surprise if even very young children fall in love with light switches and remote controls! A similar thrill can be felt as we engage in face-to-face communication with close friends, currently absent, via skype.

Distance action, remote control, and telepresence, I suggest, are at the core of what programming is about, at least experientially, because it is no longer a matter of *making things do things* by pushing them around physically. Instead, it is about signaling what we want the things to do. It involves "telling" them and giving them orders, mediated-ly. As parents say to their 2 years olds: Use words! No need to punch (or use brute force)!

Things that stand for something else, and things that stand on their own!

In a rich corpus of experiments, Judy De Loache and colleagues have shown that young children have difficulties in understanding the representational nature of objects that are interesting in themselves (De Loache, Uttal & Pierroutsakos, 1998). A scale model of a room, for example, is salient and appealing in its own right, and treated as such by a 3-year-old. In the well-known teddy bear experiment, a scale model of a room gives children information about a full-sized room that the model is meant to represent. In a preliminary phase, De Loache makes it clear to the children that the scale model is an exact mini-replica of the full-sized room, and that whatever happens in the model simultaneously happens in the room: "A big bear lives in the big room and a little bear in the little room. Whatever the baby-bear does, the daddy-bear does too". De Loache then hides the baby bear in various places in the scale model and asks the child to find the big bear "who is hiding at the exact same place in the big room". Understanding that the miniature replica stands for the larger room, the authors argue, requires that the child disentangle two functions embedded in the scale model: while it is an object in itself, it also serves as a representation of something else. Such dual representation is not constructed before age four.

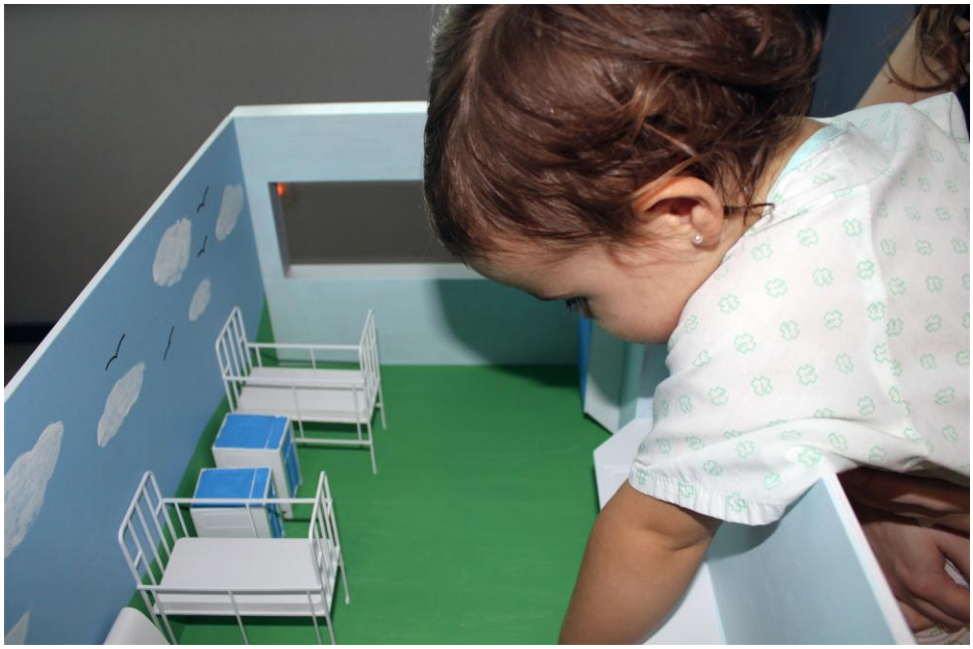

Fig.2 - *The Teddy bear is there! Source: http://senmarco.wordpress.com/2012/01/20/something-more-that-colour-project-raffle/ (public domain)*

Variations on the teddy-bear experiment further suggest that the task is, ironically, made easier if the "model" is a picture (2D) or, even better, when the symbolic relation between model and room is altogether removed. This was done in the ingenious "shrinking room" experiment, where 2-3-year olds are told that the scale model *actually IS the room — which has been shrunk by the incredible shrinking-machine*. The enactment of the shrinking operation, as unbelievable as may be, is well understood by the children, and "forces" the model-room equivalence, thus allowing for successful retrieval. According to De Loache, researchers generally agree that arbitrary symbol systems, such as numbers and letters, are difficult for young children because they bear no resemblance to their referents. What is less obvious, is that the more engaging a "representation" is for its own sake, the harder it is to treat it as something which stands for something else. And indeed one wonders, why wouldn't children take a 3D model just for what it is: a little theatre, a doll-house of sorts, a mini-stage where they can enact and play out different scenes afforded by the décor, props, and mini-figures (like teddy-bears)?

Models, simulations, microworlds

In discussing the implications of their findings for education, De Loache and colleagues express doubts as to what children may take away from watching edutainment programs, such as "Sesame Street," in which letters and numbers are being personified, and in which they talk, sing, and participate in beauty pageants. In their view, turning abstract symbols into concrete objects is likely to make their meaning less, rather than more, clear to young children (De Loache, Uttal & Pierroutsakos, 1998). Does this imply, as the authors suggest, that symbol systems, or models, ought to be more abstract, less lively, to engage learners in symbolic activities? I am not sure. Children's resistance to treating interesting objects as tokens, or go-betweens (things that stand for other things) may be a call to challenge *correspondence theories* of representation altogether (Lakofff & Johnson, 1981),

by recognizing that external representations, or models, are never copies of reality but translations. And like any translation, they transform the original. If such is the case, why not let kids be the naïve *correspondence theorists* they are, and encourage (rather than dissuade) them to treat a model (static or dynamic) not as a *simulator* but as a *stimulator* (Resnick, 1990), or a *microworld* (Papert, 1980). The difference between the two lays in the model's alleged truthfulness to reality. Simulations are generally meant to be true to real, whereas microworlds, as Papert defines them, claim their status as "alternative realities". Their purpose is not to mimic, but to bring into being and open up for scrutiny otherwise invisible mechanisms or unthinkable thoughts.

In sum, rather than debating whether representations should be more or less abstract, a more radical view is to move away from correspondence theories altogether, and to provide learners with a rich and varied palette of tools, techniques, and manipulatives, to help them capture, visualize, enact, and revisit otherwise "hidden" aspects of some intriguing phenomenon.

Machines and mechanisms – Agency, causation, delegation

Early on, children endow objects that behave with a life of their own, and treat them *as if* they were animated. This not to say that 5-year-olds believe that a computer or a robot is alive: They know it is not (Carey, 1985, Turkle, 1984). Yet in their play, the children still treat them as if they are social agents capable of initiating, sustaining, and controlling behaviours. Research on children's animism by Inagaki & Hatano (1987), Carey (1985) and Steward (1982) further suggests that children's tendency to attribute agency applies beyond 'intelligent' artefacts to include transactions among objects in general.

The most striking characteristic of children's understanding of causal transactions is that they describe the moves between interacting entities (alive or not, agent or recipient) in terms of how each impacts, or is impacted by another's, either through direct or mediated action. Note that in the case of direct action, an agent A does something to a recipient B, by *impacting it physically*, whereas in the case of mediated action, agent A signals something to B, and B acts or signals back accordingly. In both cases, agents at play tend to be animated, at least while currently active, and recipients tend to be objectified. In a chain of transactions, any particular object is by turns seen as an agent or a recipient, depending on whether it is perceived as generating an action from within (agent), or responding (recipient) (Ackermann, 1991).

A study by Ackermann and Brandes (Brandes, 1992) on children's conceptions of simple machines brings further evidence to the notion that the criteria used to determine 'machineness' is relative to a tool's ability to give back something different than what was put in the first place. In exploring elementary-school children's sense of mechanism, we asked small groups of 5 to 9 year-olds *what*, in their eyes, *makes something a machine*, and *how machines work*. We then presented individual children with small collections of images showing instances of devices with similar functionality, yet different in their source of power, level of complexity, and control mechanisms. We asked the children which of the objects were machines, and why. Examples of collections include: skateboard, bicycle, car (all used for transportation); and scissors, power lawn mower, push lawn mower (all used for cutting). Items were presented one by one.

Although children were far from unanimous as to which objects were machines, a number of regularities emerged. In session one, all groups produced *definitions by use* (A machine is something that helps you go places). Groups' ideas on how machines work revolved around four arguments: they have motors, powers, electricity, or a mechanism. In session two, individual children's groupings showed that almost everyone drew a line between machines and non-machines in terms of an object's ability to transform its input. An object, then, is a machine if it modifies what you do to it in ways that appear to make a difference. Thus for one child, *scissors* are not a machine because "it's you who cut". A *push lawn mower* on the other hand is a machine because "*you* push and *it* cuts". To the question: "what are scissors then? The child answers "a tool". For another child a *car* is a machine because "it has a motor". A *bike* is not a machine because "its you who pedal". Yet a bicycle-powered *aircraft* (as seen at Boston Science Museum) is a machine because "if you pedal and it flies...then it's got to be a machine"! In all cases, the perceived value added requires an entity capable of generating it from within. Yet the mechanism itself is mostly treated as a black box. Only upon request do children refer to it as the 'brain', the 'motor', or the 'powers'.

What does this all have to do with programming?

Media theorist and critic Douglas Rushkoff has a saying: Program or be programmed! In his view, if we don't partake in creating a culture that at least knows there's a thing called programming, then we'll end up being not the programmers, but the users, and, worse, the used. To which he adds: "Whether or not today's "creatives" (artists, designers, makers) are interested in studying the impact of technology *per se*, the learning and sharing of techniques that most people accept passively is a statement of emancipation from unidirectional tech consumption" (Rushkoff, 2010). This view is not so different from Papert's statement that computers shouldn't be used to "program" the child, but that *the child should program the computer* and, in doing [I quote]: "acquire a sense of mastery over a piece of the most powerful technology and, at the same time, establishes an intimate contact with some of the deep ideas from science, maths, and the art of intellectual model-building" (Papert, 1980).

Problem is: Like computation itself, programming is a Pygmalion. It becomes what you want it to be. To a scientist, for example, it may be a powerful tool for modeling or "simuling" the dynamic pattern of interactions at play in a complex ecosystem. To a game-designer, it may be a means to create a 3D interactive virtual habitat or an animation. And to a developmental psychologist, of which I am, the most intriguing and somewhat under-explored promises of programming lay in its ability to bring to the fore issues of control and communication between humans and machine (Ackermann, 1991).

Programming has also changed, both in its look-and-feel and nuts-and-bolts, with the developments in computing (object-oriented programming, parallel distributed computing, A-life) as well as the uses of informal 'programming' (ambient computing), and its growing popularity among non-computer-scientists.

Lastly, new materials, displays, and projection capabilities are available these days, which allow many adults—and youngsters—to "program" one way or another, which in turn bears the question: What is programming in the first place? Is it about writing code? Is it a way of thinking? Anything in-between? The answer to this question is not simple.


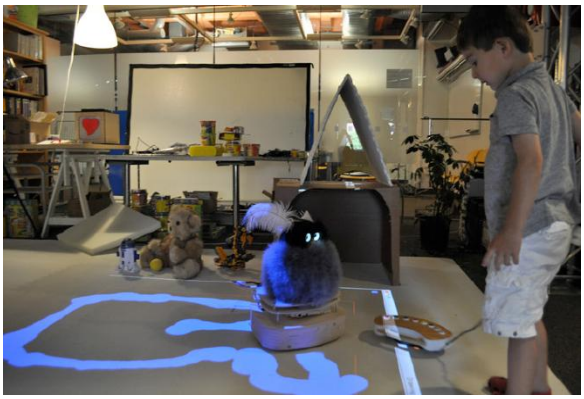

Fig. 3a/3 b *Ambient progamming. Right here. Right now. Improvise on the fly. Photo Ryan Wysort*

Programming games – Learning from the children!

Programming, at its core, is about giving instructions—or commands—to be executed by a machine. Clearly, the machine needs not to be a computer. It can be a robotic device or a set of 'smart bricks'. And the commands need not be typed on a keyboard, but can take the form of components to be assembled manually, icons to be snapped into place, and increasingly, voice, gesture, force-feedback. As Eisenberg and Buechley put it, while it is unlikely that "classical" programming will (or should) disappear, it will ultimately be one among a much larger landscape of programming styles: physical, tactile, sensually rich, athletically demanding. And as "programming" comes to suggest a different type of activity, the stereotyped portrait of "the programmer" itself will evolve (Eisenberg & Buechley, 2009. 7).

The focus in what follows is on views and uses of "programming" as a means for 1) *making things do things* (instruct a device to follow and execute orders); for 2) "*animating" things* (endow a device with a "mind of its own", teach it to "look out for itself")*;* and for 3) *"poking" things* (modulate how things act or interact by tweaking some parameters in their environment). I present settings where youngsters are asked to give and execute orders, take over control and let go of it. I draw lessons for the design and evaluation of programmable play kits for young children.

- **Programming as giving instructions: Tell it what to do!** Instructions, or directions, can be passed on verbally or cast on a piece of paper. This is usually not thought of as programming. In a program, the orders given should meet a responsive medium able to execute them. In other words, orders are encrypted as a series of operations to be read and run by a "smart" device. As a way of illustration, imagine the following scenarios, in which children tell their "smart toys" what to do.

S1 Bossing around your robotic dog. [Vignette]: *A bunch of 5-8 year olds are clapping in their hands to get a robotic dog toy to wiggle around. If they clap once, the dog wiggles its tail, if they clap twice, it wiggles its head, frantically (as if smiling), and if they clap 3 times, the dog sits down.* [Comment]: This way of bringing simple "programming" operations (in this

case if-then rule) into the environment is not unlike what Eisenberg (2009) refers to as 'ambient programming.'

S2: Telling tales with Tell-Tale. [Vignette];. *A group of 4 to 8 year olds are busy telling short story-fragment into a small hand-held voice recorder in the shape of a ball: Five kids, five recording balls of different colours, five story-bits. Once the story-bits are recorded, the kids hook together the balls to form a "caterpillar", called "Tell-tale.* [Comment]: Tell Tale is fairly silly. All it does is to play back a string of recorded sound bits, from its head to its tale. Its ingenuity lays in the fact that it allows the children to re-combine previously recorded bits to compose interesting narratives. And the kids are quick to learn how to improve their tales. Each time, they change the order of the balls and/or record new sound-bits (Annany, 2001)

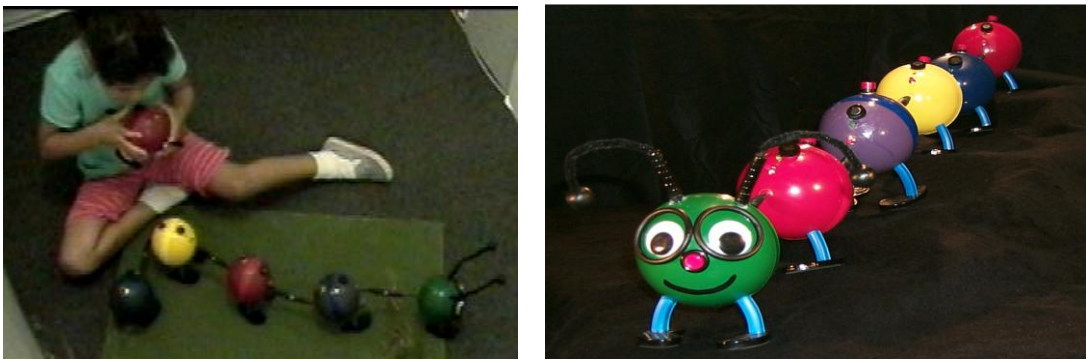

Fig 4a/4b –*Tell it to Tell-Tale!. Compose a story. Create a soundtrack. Photo Mike Ananny*

These vignettes show that while programming requires a responsive medium able to read and execute instructions, we usually don't speak of programming if a single input triggers a single response (as in ringing a doorbell or turning on a radiator). Yet we may, and many children do, if we set a thermostat to turn on the heat whenever the room temperature drops below a certain threshold, or if we set an alarm to ring at a later time, or in a different place.

S3: Setting your washer/dryer – *A typical 8 years old explains: I'm not programming if I get my coffee grinder to grind my coffee, or when I start my car but I AM programming when I use my washer to do my clothes […] because as soon as I turn the knob to start the program"it gets going and knows to do the job all the way through.* [Comment]: it remains unclear if children this age think it is THEY who are doing the programming (as they set the dial) or if, in their view, the machine was (pre)-programmed to "get itself going" as soon as they turn a knob (Ackermann, 2000, Brandeis, 1992).

In sum, programming-as-giving-instructions is best thought of as an asymmetrical transaction in which a person [the child] tells a thing [EX: a washer] to do something [like washing laundry]. And provided the instructions are understandable to that thing, it is going to do, on its own, what it was asked to do. The name of the game is: "making things do things". And like computer scientists, the children are not always clear on whether the "smarts" reside in the thing or in the person who designed, or uses, the thing.

- **Programming as lending autonomy: Make it "look out for itself".** With the advent of object-oriented and parallel distributed computing, people's views on programming took on a different tinge, which comes with its own share of underlying metaphors:

- *Metaphor 1:* From servile executant to autonomous agent: From doing things for you, the machine or smart toy is now meant do its own thing. From being a slave, it becomes a self-regulating device, or cyber-creature. Unlike its servile predecessor, it come equipped with sensors, motors, and all the in-betweens to help "it" see the world in its own unique way, have its internal reference values, and optimize "its" behaviour accordingly.

- *Metaphor 2*: One [or many] to many: The idea here is to define object behaviours in terms of attributes and methods (states, preferences, actions), and get different objects, each with their own attributes, to interact with one another to form webs, or swarms, of interconnected agents. Many surprising patterns emerge as multiple entities "unknowingly" influence each other's behaviours. Imagine the following:

S4: Critters, critters, and critters! [Vignette 1]: No computers are in sight. Elementary-school children from a Boston inner-city school are building kinetic sculptures, vehicles and creatures, out of LEGO bricks augmented with motors and sensors, plus objects that look like LEGO bricks but in fact are computational elements (flip-flops, and/or gates). One vehicle, or creature, will go towards a bright light. It has 2 light sensors (to its left and right). Each time one sensor reads "brighter", this will cause motor to turn a wheel on its side [Project Headlight, 1986]. [Vignette 2]: Even younger children, in Reggio Emilia, build and play with whimsical cyber-creatures that interact with their environment and with one another. Children build and/or influence their behaviours by acting upon their sensors and/or reconfiguring its parts. [CAB Project: http://cab.itd.ge.cnr.it, 2000]. [Vignette 3]: Children take the programming operations into the environment to drive their turtles, but this time, in the form of bar-codes on "stickers" to be read by a program reader (Eisenberg, 2009).

Our own research on children and robots indicates that interacting with artifacts that exhibit self-regulating behaviors is different from giving instructions to things that execute orders. In each case, the degree of autonomy of the artifact is different, and so are the children's responses (Ackermann, 1991). To many, undoing a creature to see what's inside is not the point. Nor are they particularly keen on building or programming a bot. Instead, they spend much time finessing their dance with a creature and, in doing, they experience the pros and cons of shared or distributed control. The purpose is to converse rather than construct, to bond rather than rule, and to feel connected rather than in charge.

- **Programming as "poking": *Make do* with what's there and revamp!** More than in the past, today's computational tools and materials encourage people to program in a weak sense, by modulating rather than making, i.e., by tweaking existing programs rather than having to write their own. Creators can import entire chunks of text, image, and sound [including code], which they then re-combine as they please. In other words, no need to start from scratch or to write a single line of code: You just borrow what's there, and you "remix". This shift from creating to modulating existing behaviours has important implications for education. It also gives rise to new forms of playful interactions between people and animated toys.

S7: Assemblages *[Vignette] A connected-classroom and a bunch of 8 year olds, sitting in front of their laptops. Children are getting ready for a class project on Egypt (to be shared online). They surf on the web until they hit some page they like. They import the page, or parts of it, and use it as a template that they then "massage" until it no longer resembles the "found" original or inspirational seed, but becomes their own*. [Comment]: This found art

approach to writing generates big controversies among educators who wonder if children (by shamelessly borrowing and tweaking) are still writing, let alone be the authors of their writings. My contention is that, provided the borrowers "massage" a template long enough, they indeed are writing! It is not exaggerated to say that there is not such a thing as starting from scratch. The same can be said of programming (Ackermann, 2011)

S8: Playing with the elements *[Vignette] Children in France have forever gathered around circular water basins (common in public parks) to play with tiny colorful toy sailboats and long thin wooden rods (Fig. 5a) that the children can rent. It is easy to imagine the spectacle as children, each with their rod, attempt to launch, direct, track, loose touch, and welcome back their boat! In recent years sailboats been replaced by motorboats and the rods by remote controls* (Fig.5b). [Comment]: Experientially, it feels very different to steer an obedient toy motorboat using remote-controls, than to tame a capricious wind-driven little sailboat by means of a rod. Poking sailboats is more like flying kites: You are not the only master on board. You play with the elements, Steering motorboats is more like bossing around a robotic dog on a smooth ground. Not much gets in the way of your will.

Fig. 5a– *Toy sailboats and rods, Paris.*

Fig.5b- *Motorboats Cannes. Photo by author*

To conclude, different manners of controlling and influencing behaviors (making things do things) are embedded in the notion of "programming" (in a weak sense). And people's own definitions of programming, and views on the benefits of programming, depend in great part on their unspoken object-relational affinities.

In what follows, we'll see that people, depending on circumstances or personal preference, may be more inclined to favor one approach (one way of making things do things), over the other.

Why learn to program?

If, as suggested, programming is about giving instructions, lending autonomy, and modulating existing behaviors, the question remains: What's in it for the children? Why should preschoolers do it? In the light of the discussion so far, I can think of at least three reasons worth considering:

- **Mastering things: take over / let go / take over -** Through giving instructions, young children gain mastery over their world. They create and control things to execute their orders. They set them in motion, make them do things, and "boss them

around". How could this not satisfy a 3 years old's craving for omnipotence! At the same time, by giving orders to an artefact smart enough to execute them, the children incidentally learn to let go and to delegate, and delegation entails distribution of control. This happens because as soon as the artefact executes a child's orders, it also starts acting on her behalf, by taking on a part of the job.

In a playful way, the child can explore issues of task sharing (who does what for whom) and, in doing, learn about the pros-and-cons of taking over versus letting go, both so crucial in any type of transaction, be it with people or with things. Besides, even the most obedient of artifact, like Papert's Logo Turtle, is bound to behave unexpectedly (be non resilient) if the commands the child enters are unclear, i.e., unintelligible to 'its' kind of mind. In playing turtle, children are given an occasion to learn to state explicitly what they want, in a language understood by their interlocutor.

- **Animating things: create / animate / interact -** By building and playing with things that act *as if they had a will of their own*, young children learn about the ways in which animate and inert objects regulate their behaviours, and how they interact with one another. Teaching things to "look out for themselves" and watching them do 'their' things is enjoyable because, beyond taking orders, the creatures have now gained autonomy. They can [be made to] follow light, avoid contact, or dance with one another. And it is fun to watch them and to enter in the dance with them.

In a playful way, the child learns to distinguish between self-driven and other-induced courses of action, between inner- and outer locus of control. She interacts with new forms of intelligence, different from her own and thus gains insights into what it means (and takes) to be "animated" or "smart", for a person and a thing.

- **Modulating things: take it as is / tweak it / let it be -** More than in previous generations, today's children are *bricoleurs*: a new bread of *makers, hackers and hobbyists* eager to gather, collect, create, and trade things and *make do with what's at hand* [2]. They repurpose (remix) the stuff they find, endowing it with a second life or extra "powers". They like to engage in *digital* crafting and fabrication. If given a chance and provided appropriate support, today's kids won't merely consume and dispose. Instead, they will create and recycle. They will care! (Ackermann, 2011)

In a playful way, the child learns to distinguish between recycling, starting anew, and restoring, or adding value to what's already there. It is in part today's children's confidence in—and knowledge about—how to fix and mend things, together with a belief in the benefits of iteration (layering, refining), afforded by computational devices, which hold the potential to bread a new culture of crafting.

Who likes to program? What's in it for young children?

Not all children like to program. Not all programming feels the same. And the very notion of programming itself, as debated among experts, is changing as we write.

[2] A bricoleur is a Jack-of-all-trades ("Bastler" in German). Adept at many tasks, he likes to put existing things together in new ways. The Engineer, in contrast, starts from scratch, with an outline, and plans ahead.

Some children may (on occasion and depending on their personal style) do anything to be in charge, while others won't mind to assist, relinquish control, delegate, or negotiate. Some get a kick out of guessing and planning ahead (i.e., write procedures), while others like to make up their mind as they go (i,e, write step by step commands). Others yet, the bricoleurs, like to compose with what's there (i.e. borrow and edit code). In spite of these differences, it is fair to say that most children, if given a chance, are thrilled to create things, or make stuff, and to bring their creations to life, by giving them 'extra powers'. What changes is the amount of building or "dancing" involved, the metaphors they draw from, the quality of the materials at hand, and the play scenarios they get excited by. For very young children, programming as modulating existing behaviors may be a way to go, although one wonders: Is this still about programming?

Fig. 6a/6b - *Very young children "programming" together. Olpc.MIT. Photos Edith Ackermann*

A program, we have seen, can be more than one thing and not all programming feels the same. Many new materials, settings, and display surfaces are at people's avail, these days, which make programming a far more informal, approachable, and natural activity than before. As Eisenberg and Buechley write: "On the one hand, a variety of traditional materials –fabric, paper – can now be employed as the background substrate for programmable artifacts and displays; that is, it is possible to work with *programmable materials*. In a similar vein, one can devise means of placing small, informal "chunks" of programs within physical environments, where they may be read or executed by mobile computational devices –a notion that we refer to as *ambient programming*[3] (…) Finally, there are novel types of *display surfaces* that may be used as the backdrop for relatively unexplored styles of programming" (2009, 1-2). These changes in turn inform the terms of the debates about the potential of programming for young children.

Our own observations of children and adolescent's uses of sensors, actuators, smart bricks, and circuits (in different STEM, and STEAM workshops) confirmed, time and again, that the materials used and the types of activities proposed have strong built-in 'affordances', which need to be considered. For example, LEGO bricks favour orthogonal structures. One must work hard to make anything curvy. Also the do-undo-

[3] The idea of "ambient programming" suggests that programs can be constructed in informal, moment-to-moment ways. One might alter the "program" by messing about with cards upon the floor, changing their positions and putting down new cards.

redo quality of many construction-kits favours *tinkering* over *crafting.* A second type of bias occurs when designers or educators impose their own limiting views on *what should* be built, and *how*. Different play scenarios excite different minds. In order to cater to personal, gender-related, and culturally related preferences, my best advise to this day is: Offer rich and diverse materials and imagine different play scenarios that may capture different kids' imagination, and they'll do the rest…

Lastly, I echo Seymour Papert's own teachings when he advised never to teach children to program for the sake of programming. Instead, he said: Use the knowledge of programming to create contexts where other play and learning can happen. Children will engage in programming if they get something out of it right now –not later *when they'll grow up*! And this, to Papert, doesn't mean that it won't take much effort to become good at what they are doing. The implication is rather that "hard fun" is challenging to most children who, we know, can spend hours on something, when genuinely interested. Thus, the question is not "what is the effect of programming, or using computers, on learning". Instead, we should ask: Can computational tools provide venues for learning and play, for exploring, expressing, and sharing ideas, in ways otherwise difficult.

Acknowledgements

I thank Reiko Sakurai and her colleagues from the Child Research Net, Japan, for giving me the opportunity to post on their "wall". This is a priceless privilege. I thank Chronis Kynigos for his insights and encouragements to write this paper, and the organizers of "Constructionism 2012" for bringing us together in Athens. I thank Mike Ananny, Florent Aziomanoff, and Ryan Wysort for the permission to use their photographs, and Seymour Papert, Mike Eisenberg, Leah Buechley, and Jose Valente for their thoughts on "programming" for children. I am especially grateful to the children with whom I have had a chance to play, and to Aaron Brandeis with whom I did much of the research on children and machines, at MIT, in the early nineties.

References

Ackermann, E. (2011). Minds in motion, Media in transition: Growing up in the digital age. Areas of change. *Child Research Net*. Japan **http://www.childresearch.net/papers/digital/2011_01.html**

Ackermann, E (2005). Playthings that do things: A young kid's "incredibles"!. *Interaction Design and Children*. **I**DC 2005, Boulder, Colorado, USA, June 8-10, 2005. Copyright 2005 ACM 1-59593-096-5/05/0006...$5.00.

Ackermann, E. **(**1999). Enactive representations in learning: Pretense, models, machines. (Bliss, Light, & Saljo, Eds.) *Learning Sites: Social and technological Contexts for learning* Elsevier: Advances in learning and Instruction. p. 144-154.

Ackermann. E. (2000) Relating to things that think: Animated toys, Artificial creatures, Avatars. *Play of Ideas and ideas of Play*. I3 Magazine: The European network for intelligent information interfaces. Volume. 8, p. 2-5, July

Ackermann, E. (1991). The Agency model of transaction. (Harel & Papert, Eds.) *Constructionism*. Norwood, NJ: Ablex Publishing Company, pp 367-379.

Annany, M. (2001) Teel-Tale. Doctoral Thesis. The MIT Media Lab. Cambridge, MA.

Brandes, A. (1992) Children's ideas about machines. Paper presented at the annual meeting of the American Educational Research Association.

Carey, S. (1985). *Conceptual Change in Childhood*. Cambridge: MIT Press.

De Loache, J., Uttal, D., & Pierroutsakos, S. (1998). Waiting to use a symbol (Demetriou, Ed) *Cognitive Development: New trends and questions.* Special Issue of Learning and Instruction, Volume 8. 4, 325-341.

Eisenberg, M, & Buechley, L. (2009). Children's programming, reconsidered: Setting, stuff, and surfaces. *Proceedings IDC 2009*, June 3–5, Como, Italy: Copyright 2009 ACM 978-1-60558-395-2/09/06.

Inagaki, K. & Hatano, G. (1987) Young children's spontaneous personification as analogy. *Child Development*., 58

Lakoff, G. & Johnson, M. (1981) *Metaphors We Live By*. Chicago University Press.

Papert, S.(1980) Mindstorms: Children computers and powerful ideas. New York: Basic Books.

Resnick, M. (1990) Turtles, Termites, and Traffic Jams: Explorations in massively parallel microworlds. Cambridge, MA: MIT Press.

Rushkoff, D. (2010) Program or be Programmed: Ten Commands for a digital Age.

Steward, J. (1982) *Perception of animacy*. Doctoral Thesis. University of Pennsylvania.

Turkle, S. (1984). *The Second Self*. New York: Simon & Schuster.