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

Edith K. Ackermann, *edith@media.mit.edu*
School of Architecture. Massachusetts Institute of Technology, Cambridge, MA., US

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 ever younger children are increasingly "expert" gamers, tweeters, information-seekers, and digital bricoleurs. Often 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 doomed 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); 2) "*animating" things (*endow them with a mind of its own, teach them to "look out for themselves"*); *and 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.*

Keywords

Program, model, object-relation, agency, control, animate, appropriation, smart" tools, 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 role of *new media literacy* (literacies *beyond print*) would be today's equivalent of promoting illiteracy. What is less clear, to this day, is the status of programming itself (beyond modelling and simulating), 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, and why, in order to satisfy 21st century skills requirements? What's there to be gained in the first place? And what's in it for the children?

Programming is many things to many people, and not everyone agrees on its potential for learning. Understanding today's children's ways of navigating and blending physical, virtual, and digital, and using ICT provides a unique window into gauging the pros-and-cons of programming in the 21st century. It does so by challenging our own assumptions on what it means to be smart, literate, or creative, athus allowing us appreciate the natives' strengths before we advise on what they'll be missing, if left on their own (Ackermann, 2011).

The purpose of this paper is to discuss the intricacies between modelling, simuling, and

programming from an experiential perspective. Building on research with, and observations of, children, we explore issues of agency, control, and interaction styles, or relational preferences, as played out in the creative and critical uses of "smart" tools: in this case, any artefact or device, physical or virtual, that is either seen as being responsive and inner-driven, treated as if it were, or made to do things on its own. The focus is on how children, and adult-experts, use and think of "programming" as a means for exploring and optimizing the interplay between a human (usually themselves) and unequally responsive, surprising, or reliable devices.

I discuss why exploring the "logic" of gives and takes, through interacting with, relying on, or controlling "smart" toys, or endowing stuff with smarts, can enrich our experience and understanding of "programming" (in the broad sense of "making things do things). I present settings in which youngsters are invited 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. Clearly, 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 elicit new ways of relating, and are used in different ways (Turkle, 1984, Ackermann, 2000)

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". It is their believable artificiality and "forgiving" nature that make it possible to explore, enact, and work through issues of identity formation 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, or at a later time. So, 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 feel the thrills of making 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, away from home, via skype.

Distance action, remote control, and tele-presence, I suggest, are at the core of what programming is about, at least experientially, because it is no longer just a matter of making things do things by pushing them around physically. Instead, it is about signaling what it is we want them to do. It is about "telling them", giving them orders, or instructions, mediated-ly. As parents say to their 2 years olds: Use words! Don't just 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, DeLoache 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..

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, reserachers 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 *representational* may be 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 their truthfulness to reality. While simulations are generally meant to be true to real, 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 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 behaviour, 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' are 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 spresented 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 an input in significant ways. An object, then, is a machine if it modifies what you do to it in ways that make a difference. Thus for one child *Scissors* are not a machine because "it's you who cut". A *push lawn mower* 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 value added requires an entity capable of generating it from within. Yet the entity itself is often 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 own 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 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, paper computing), and its growing popularity among non-computer-scientists. New materials, display and projection capabilities are at people's avail allowing many adults—and youngsters— these days to "program," one way or another, which makes one wonder: 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.

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 read and 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 scenario 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 it 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 recording device 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 bits, from its head to its tale. Its ingenuity, however, lays in the fact that it lets the users in, and re-combine previously recorded story bits to compose more interesting narratives. And the kids are quick to learn to improve the tale. Each time, they change the order of balls and/or record a new story-fragment. (Annany, 2001)

These vignettes show that while programming requires a responsive medium able to read and execute a set of instructions, we do not usually speak of programming if a single input triggers a single immediate 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 - *To a typical 6 to 9 year old: I'm not programming if I "tell" my coffee grinder to grind my coffee, or when I start my car but I AM programming when I set my washer to soak, wash, and rinse my cloth"[…] because even if I just turn a knob to start the program "it knows to do the job all the way through".* [Comment]: it is not always clear to children this age if THEY are the ones who program the machine, or if the machine itself is programmed to "respond and get itself going" as they turn a knob or pushing a button (Ackermann, 2000, Brandeis, 1992).

Programming-as-giving-instructions is best thought of as a dialogue-in-action in which a person [the child] tells, or teaches, a thing [EX: a washer] to do something [like washing laundry] on her behalf. In other words, a child delegates a job to a thing 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: "make things do things". Incidentally, our own research on children and machines indicates that for children too programming is about getting a machine, computer or toy, to help you do things that require smarts. And like computer scientists, the children are sometimes unsure if the "smarts" reside in the machine itself or in the person who designed, or used, the machine

- **Programming as lending autonomy: Make it "look out for itself"**. With the avent 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. It is gaining autonomy. Unlike its servile predecessor, it come equipped with sensors, motors, and all the in-betweens to help "it" see the world its own way, have its internal reference values, and optimize "its" moves accordingly.
- *Metaphor 2*: One to many: The idea here is to define an object's behaviors in terms of

attributes and methods (states, preferences, actions), and then thave different objects, each with their own attributes, interact with one another, to form wide webs, or agencies, of interconnected agents. Many emerging patterns form as multiple entities interact with one another. Imagine the following scenarios:

S4: Critters, critters, and critters! [Vignette 1]: No computers are in sight. Elementary-school children from a Boston inner-city school are building animated sculptures, vehicles and creatures, out of LEGO bricks augmented with motors and sensors, plus objects that look like LEGO bricks from outside but in fact are computational elements (flip-flops, "and" gates). One vehicle, or creature, will go towards a bright light. It has 2 light sensors to its left and right: If one is brighter, this will cause a motor to turn a wheel on its side. [Project Headlight. LEGO Logo workshop, 1986]. [Vignette 2] : Even younger children, in Reggio Emilia, build and play with funky cyber-creatures that interact with their environment and with one another. Children can build and/or influence their behaviors 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).

Research on children and robots suggests that interacting with artifacts that exhibit self-regulating behaviors is different from giving instructions to things that executes orders. In each case the degree of autonomy of the artifact is different, and so are the children's reactions (Ackermann, 1991). To many, undoing a creature to see what's inside the black box is not the point. Instead, they focus on optimizing the dance with a creature and, in doing they experience and play out the trade-off and potential of mutual influence, and shared control. The purpose is to converse rather than construct, to attune rather than break down, to empathize rather than analyze.

- **Programming as "poking": Don't start from scratch. Make do with what's there !** More than in the past, today's computational tools and materials encourage people to program in a weak sense, by modulating rather than making, or tweaking existing programs, without ever having to produce a single line of code. Creators can import 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: You borrow what's there, and you " "remix". This shift from creating to modulating existing behaviours has important implications for education.

S7: Assemblages *[Vignette] A connected -classroom and a bunch of 8 year olds, sitting in front of their laptops. Each child is busy "writing" for a class project on Egypt (to be shared online). How do you think the kids proceed? Well here's what they do: they surf on the web until they hit some page they really like. They import the page, or parts of it, and they 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 thse days (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 writing on a blank page. The same can be said of programming (Ackermann, 2011)

From a psychological perspective, the difference is significant between using programming to instruct and obedient contraption (as in Turtle Geometry) and to influence the course of a self-regulating device by intervening in [as a part of] its own environment (as in mindstorms). And so is the difference between 'dancing" with an artificial partner and bossing around a tech-toy. In what follows, we'll see that some people are more inclined to favor one approach over the other.

Why learn to program?

If, as we 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 3 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 boost a 3 years olds' craving for omnipotence! At the same time, and ironically, by giving orders to an artefact reliable and smart enough to execute them, children also learn to let go and to delegate. And delegation entails distribution of control because as soon as the artifact executes a child's orders, it also acts 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 a great deal about the pros-and-cons of taking over versus letting go, so crucial in any type of transaction, be it with people or with things. Besides, even the most obedient of artificial critters, 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 unique occasion to learn to state explicitly what they want.

- **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 interact with one another. Teaching things to "look out for themselves" and watching them do 'their' things is enjoyable because, beyond executing 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 enter the dance with them.

In a playful way, the child learns to distinguish between self-driven and other-induced, between inner- and outer locus of control. She interacts with new forms of intelligence, different from her own and gains insights into what it means 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 generation, today's children are often *bricoleurs* instead of planers. They are a new bread of *makers, hackers and hobbyists* eager to gather, collect, create, and trade objects, and ready to *make do with what's at hand* [1] They repurpose (remix) the stuff they find, endowing it with a second life or extra "powers". And as they grow older and perfect their technical skills, they often engage in the art of *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 adding value to what's there. It is in great part today's kids' 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.

[1] A bricoleur is a jack of all trades or "Bastler" in German. Lévi-Strauss depicts the "bricoleur" as adept at many tasks and putting existing things together in new ways. The Engineer, in contrast, usually starts from scratch, with an outline, and plans ahead

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 notion of programming itself, as debated among experts, is changing as we write.

Starting with the children, some may (on occasion and depending on their personal style) do anything to feel in charge, while others won't mind to delegate, or negotiate. Some will get a kick out of guessing and planning ahead (i.e., write procedures), while others prefer 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, when given a chance, will be happy to create things, or make stuff, and bring their creations to life, i.e give them 'extra powers'! Our own research indicates that children's apprehension of computational devices is at usually both instrumental and relational (Ackermann, 1991). What changes is the amount of building or "dancing" involved, the metaphors they draw from, the quality of the materials, 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?

A program, we have seen, is 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*[2] (…) 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, materials, smart bricks, and circuits (in different robot, 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 taken seriously. For example, LEGO bricks favor orthogonal structures. One must work hard to make anything curvy. Also the do-undo-redo quality of most computational construction-kits favours *tinkering* over *crafting*. A second type of bias occurs when designers and 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 a range of play scenarios that may capture different kids' imagination, and they'll do the rest…

Lastly, I wish to echo Seymour Papert's own teachings when he advised not to teach children to program for the sake of programming. Instead, he said: use the knowledge of programming to create contexts where other playful learning can happen. Children will engage in programming if they can get something out of it right now –not later *when they'll grow up*! And this, to Papert,

[2] The idea of "ambient programming" suggests that programs can be constructed in informal, moment-to-moment ways. One might alter the "program" shown in Figure 9 by physically messing about with cards upon the floor, changing positions and putting down new cards.

doesn't imply that it won't take much work, or effort, to become fluent at what they are doing. The implication is rather that "hard fun" is usually more challenging to the children who, we know, can spend hours on something, repeatedly, when they are genuinely interested. Thus, the question is not so much " what is the effect of programming, or using computers, on learning". Instead, we should ask: Can computational tools provide new venues for learning and play, for exploring, expressing, and sharing ideas, otherwise impossible.

Acknowledgements

I thank Chronis Kynigos for his time, insights, and encouragements to write this paper. I am grateful to the organizers of "Constructionism 2012" for bringing us together this summer. I thank Seymour Papert, Mike Eisenberg, Leah Buechley, Mitch Resnick, Jose Valente, Sherry Turkle, and my friends and colleagues from the extended learning-and-epistemology tribe for their thoughts on children's experience of "programming". I am especially grateful to the children with whom I have had a chance to play.

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/RESOURCE/RESEARCH/2011/ACKERMANN.HTM**

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. **http://youtu.be/imv3pPIUy**

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

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