Playing at Work

Seymour

The expected division of labor between kids and adults is for the former to play and the latter to work. This is particularly true of their relationship with technology, which creates toys for kids and tools for adults. But the inventiveness of children has led to a historical blurring of the distinction between toys and tools for invention, culminating in the integration of play and work in the technology for personal fabrication. The original inspiration and instigator for bringing these worlds together was Seymour Papert, a mathematician turned computer pioneer.

Seymour first encountered computers by a fortuitous accident. He had met MIT's Marvin Minsky, considered to be the father of the study of artificial intelligence (AI), at a conference in the late 1950s. Their meeting was memorable because they had, without having heard of each other, submitted essentially the same paper to the conference. They were both thinking about how to mathematically model the way the brain reasons, in order to better understand it. Both approached this question by abstracting the essential features of networks of neurons; the similarity of their approaches led to their joint authorship of a book titled *Perceptrons*, which laid the foundation for generations of study of neural networks. But their meeting also led Seymour in a very unexpected direction when Marvin invited him to come to MIT.

Seymour had been in Geneva, working with the famous Swiss psychologist Jean Piaget. Piaget articulated the "constructivist" approach to learning, recognizing that children learn through experimentation much more than by recitation and repetition. They're really little scientists doing endless experiments, an admirable trait that gets trained out of them in a conventional education based around a rigid curriculum.

In 1963 Seymour traveled to MIT expecting to extend his study of learning by developing better models of artificial intelligence. Seymour arrived, but Marvin didn't: there was no sign of Marvin at the appointed time and place (it later turned out that Marvin had remembered the wrong day for their first meeting). This fortuitous mix-up left Seymour sitting alone in a room with a curious-looking machine that turned out to be one of the first minicomputers, Digital Equipment Corporation's PDP-1, serial number 2. This was the commercial descendant of the earlier experimental TX-2 computer developed at MIT, and as part of the licensing deal the DEC agreed to provide a preproduction model of the PDP-1 for use at MIT.

At that point in the history of computation, even an hour of mainframe time was a precious commodity. But the PDP-1 was sitting idle because there was not yet a demand for individual computer use. No one was sure what it would be good for. Mainframes might be valuable for running accounting systems or tracking inventory, but who wants to do that in their free time?

Since the computer was available, and Marvin wasn't, Seymour started fiddling with it. He learned how to program it, planning to use it to generate optical illusions with random dot patterns as a way to study how human perception works. His success doing that opened his eyes to an even greater possibility: in the programmability of the computer, he had found the ultimate sandbox for kids to play in. If children could get the same access Seymour had to a machine like the PDP-1, they could play with abstract concepts with the same ease that they play with a lump of clay.

There were two problems with realizing this vision. The first was the PDP's machine language that Seymour had to learn to program it. We'll meet machine language in the next chapter; it's about as unfriendly as a language can be. Seymour needed to develop a language that both kids and computers could understand. At that time, the new language of choice for computer scientists at MIT was LISP. A bit of the LISP code for Emacs, the editor I'm using to write this book, looks like this:

```
(if (= (car n) 0)
  (cons 0 s)
 (cons 1 (cond
       ((not (listp (cdr n)))
        (list 'vconcat (cdr n)))
       ((eq (nth 1 n) 'list)
        (cons 'vector (nthcdr 2 n)))
       ((eq (nth 1 n) 'append)
        (cons 'vconcat (nthcdr 2 n)))
       (t
        (list 'apply '(function vector) (cdr n)))))))
```

For anyone, like me, whose brain does not work like a computer scientist's, this is a foreign language. But it is based on an elegant model of programs operating on lists of data, which led Seymour to abstract its essence into the LOGO language aimed at kids. LOGO works much like LISP, but unlike LISP, it's written in words that mortals can read.

The second problem was providing kids with access to computers; the PDP-1 wasn't even on the market yet. Seymour solved this by employing the newly developed capability of computers to do more than one thing at a time. Time-sharing made it possible for a remote user in a classroom to dial into MIT and run a program, with the computer switching between multiple users' programs so fast that it appears to any one of them to respond in real time. At the time, this was harder than it sounds; the development of "high-speed" telephone modems that could send data at three hundred bits per second was a cause for great celebration (today's cable modems run at millions of bits per second).

Teaching turtles

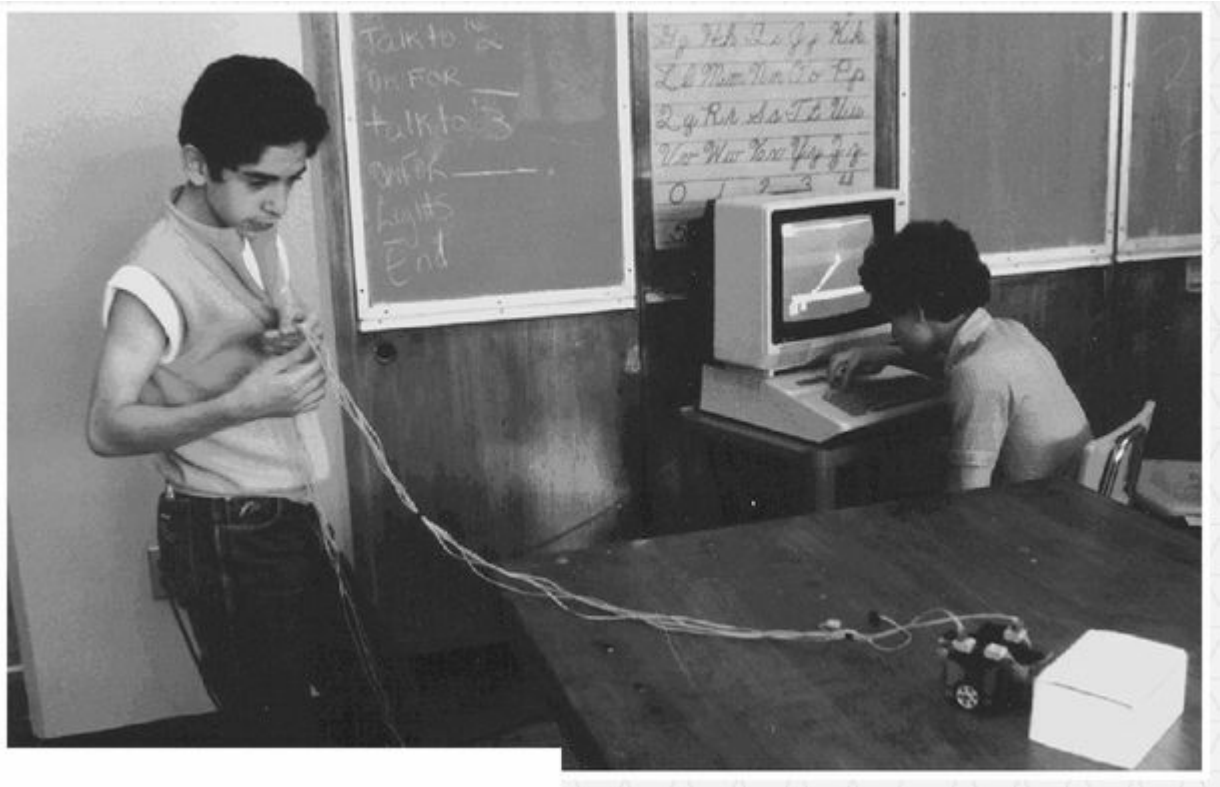
Seymour tried an experiment, connecting kids in an inner-city Boston classroom to remote computers. This worked well with older children. Instead of Seymour's original plan to develop models of AI, he had the kids do it. Just like AI researchers at MIT using LISP, the children used LOGO to explore the consequences of rules they wrote for how the computer should respond to them. But that was too abstract to engage younger kids, below around fifth grade. Because physical manipulation is essential to how they learn, Seymour wanted to provide physical devices they could use to interact with the computer.

A staple of early AI research was a robotic "turtle," a little vehicle controlled by an electronic nervous system. Seymour developed a turtle that could connect to a remote time-sharing computer. A program's output could be embodied in how the turtle moved and reacted to its surroundings. Along with the turtle a number of other physical interfaces were developed, including a box of control buttons, and a device that could read instructions on cards. (The latter was developed by a precocious undergrad, Danny Hillis, who went on to become a precocious adult architect of supercomputers.)

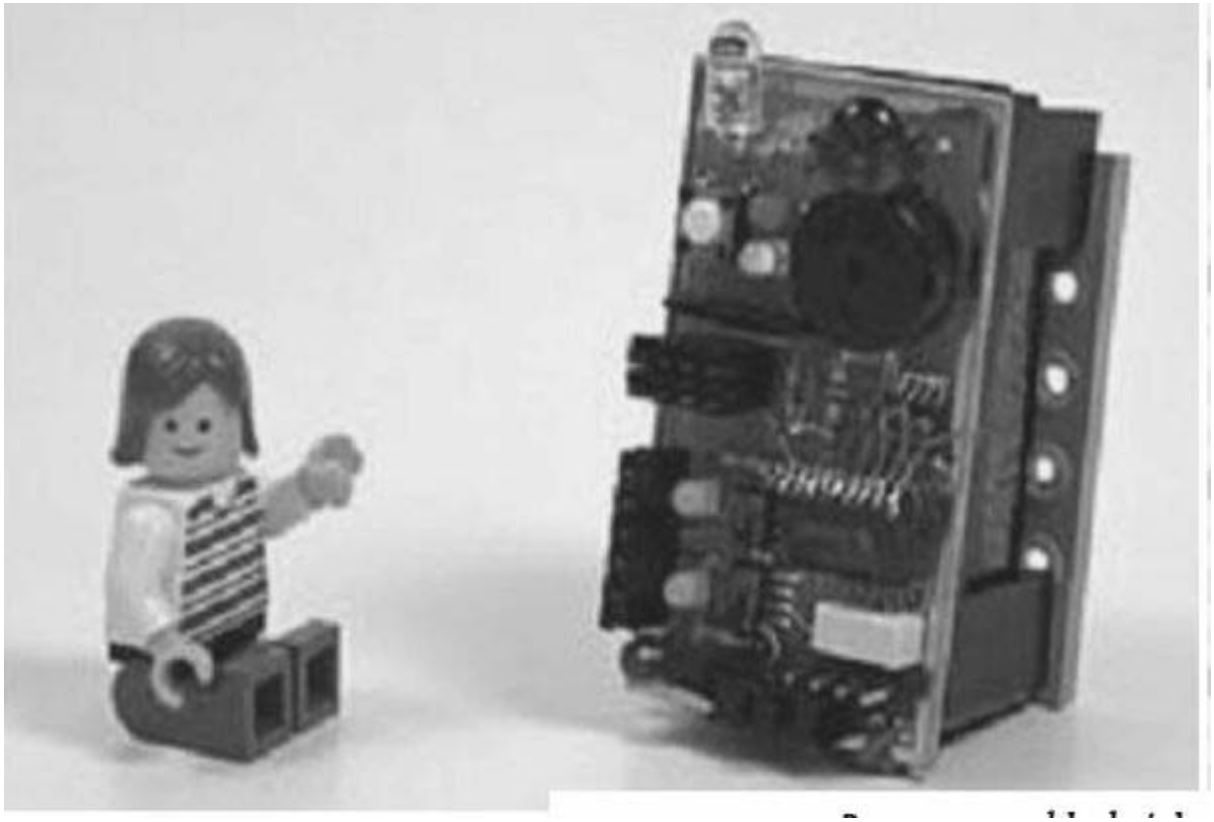
These capabilities were far ahead of what was available to the typical corporate computer user, who had to wait years before this kind of real-time computer interaction became available in mice, pens, plotters, and other input-output devices. For kids, the innovation in the physical form of their computer interfaces paused when computers developed to the point that turtles moved onto the screen as computer graphics (around 1970). This was not as compelling as an

(artificially) alive turtle moving in the real world, but it made LOGO programming available to anyone with access to a computer. Versions of LOGO migrated from minicomputers to the emerging personal computers, from the TI-99 to the Atari 400 to the Apple II to the IBM PC.

While this was happening, a new collaborator appeared, thanks to the BBC. Kjeld Kirk Kristiansen, the head of LEGO, saw a BBC news program about the turtles, and is reported to have said, “Hey, there’s someone who thinks like us!” At the time, LEGO was not yet the icon it’s become. LEGO was started by his grandfather, Ole Kirk Christiansen, an out-of-work carpenter eking out a living in Denmark in the 1930s making wooden toys. After the interruption of World War II, Ole Kirk decided he liked making toys so much that he went back to it. One of his products was a wooden truck. When moldable plastics came along, he used them to add bricks as a load for the truck. Sales of the truck took off, but it was bought for the bricks, which were more fun than the truck. The rest is business history.



Controlling computers



Programmable brick

If a computer could connect to a turtle then it could connect to LEGO bricks, allowing kids to add functional behaviors to the physical shapes they were making. Seymour's successors at MIT, led by Mitch Resnick, worked with LEGO on developing kits to interface LEGO sensors and motors to programmed LOGO worlds in PCs. This was promising, but playing with the bricks still required a desktop computer.

Meanwhile, computing continued its evolution from microprocessors powering PCs to even smaller microcontroller chips that were complete simple computers that could be embedded in products. Mitch and his colleagues Fred Martin, Brian Silverman, and Randy Sargent used a microcontroller to develop a LEGO/LOGO controller about the size of a child's wooden block. This became the model for the Mindstorms robotic construction kit (named after a book by Seymour). That group, along with Bakhtiar Mikhak and Robbie Berg, continued this evolution, shrinking the processing down to the size of a single LEGO brick.

Kids loved this, in a number of often unexpected ways. In an early test in a fourth-grade class in inner-city Boston, the boys quickly caught on to the computing bricks, using them to make all sorts of vehicles. The girls weren't so keen on the computing part, choosing to stick with the original bricks to design houses. But they were eyeing what the boys were doing, and discreetly started adding features to the houses, like lights that could be turned on and off. Then they started programming patterns for the lights, and from there started inventing robotic appliances. They were hooked, and didn't even realize that they were learning how to design engineering control systems.

David

In 1999 I wrote a book, *When Things Start to Think*, on the emerging technology and implications of computing moving out of conventional computers and into everyday objects, from LEGO bricks to footwear to furniture. I was pleasantly overwhelmed by the detailed letters I received from grown-up engineers, discussing their own inventions, past, present, and future, ranging from eminently practical to utterly impossible. One of these stood out as being particularly well written. It began with a thoughtful critique of the book, then veered off in an unexpected direction:

> True or False:
> A school bus in 2045 will:
>
> a. have sensor-controlled gyroscope stabilizers.
> b. be able to detect rule-breakers and their names, then report them to the driver.
> c. have thought-controlled windows.
> d. store in memory frequented destinations (such as the indigenous school), then drive the bus

automatically (Now there's an AUTOmobile!) to those destinations.

Intrigued, and puzzled, I read on to find closing observations including "The future of the future looks bright." I then nearly fell out of my chair when I read the signature: "Cordially, David (Just turned 8 years old, about to enter 9th grade)." Not sure if this was serious or an odd kind of joke, I wrote back a tentative reply, which was followed apparently instantly by a seven-page letter from his mother, Athena, which roughly translated to "Help!!!!" She was home-schooling David because regular schools couldn't keep up with him. He was indeed only eight years old, and her problem was that he was ready for college and she wasn't sure what to do with him.

Tea technology

This began a lively correspondence with both David and Athena. Through our exchanges I learned that David had been saving up to buy a Mindstorms kit, because his ideas for inventions were racing far ahead of his ability to realize them. I arranged for him to get one, which he quickly devoured, mastered, and put to use in projects.

I thought of David again when that same year the White House asked me to put together a panel on the future of science and technology for the millennium celebrations. If our future is our children, and our technological future lies in their inventions, who better to give a peek at that than an eight-year-old? Along with an assortment of older students and grown-up scientists I added David to the panel, unwittingly making a terrible mistake.

All sorts of impressive technology was on display, from real-time atomic manipulation to embedded intelligence in paper, clothes, and furniture. David brought a Mindstorms invention he made for his mother to address her frustration with her cup of tea being either too hot or too cold. To ensure that her tea was just right, he made a machine that mixed in cold water, stirred the tea, and announced when it had cooled down to the ideal temperature.

The mistake I made was not putting David at the end of the program: He stole the show. After him, it was hard for anyone to present anything else. David's presentation was smart, funny, relevant, and moving. He used his tea server as a springboard to discuss how broader access to the tools for technological development could help humanize technology, better meeting society's needs by involving individuals in posing and solving their most pressing problems. The future of the future is indeed bright in the hands of young inventors like David.

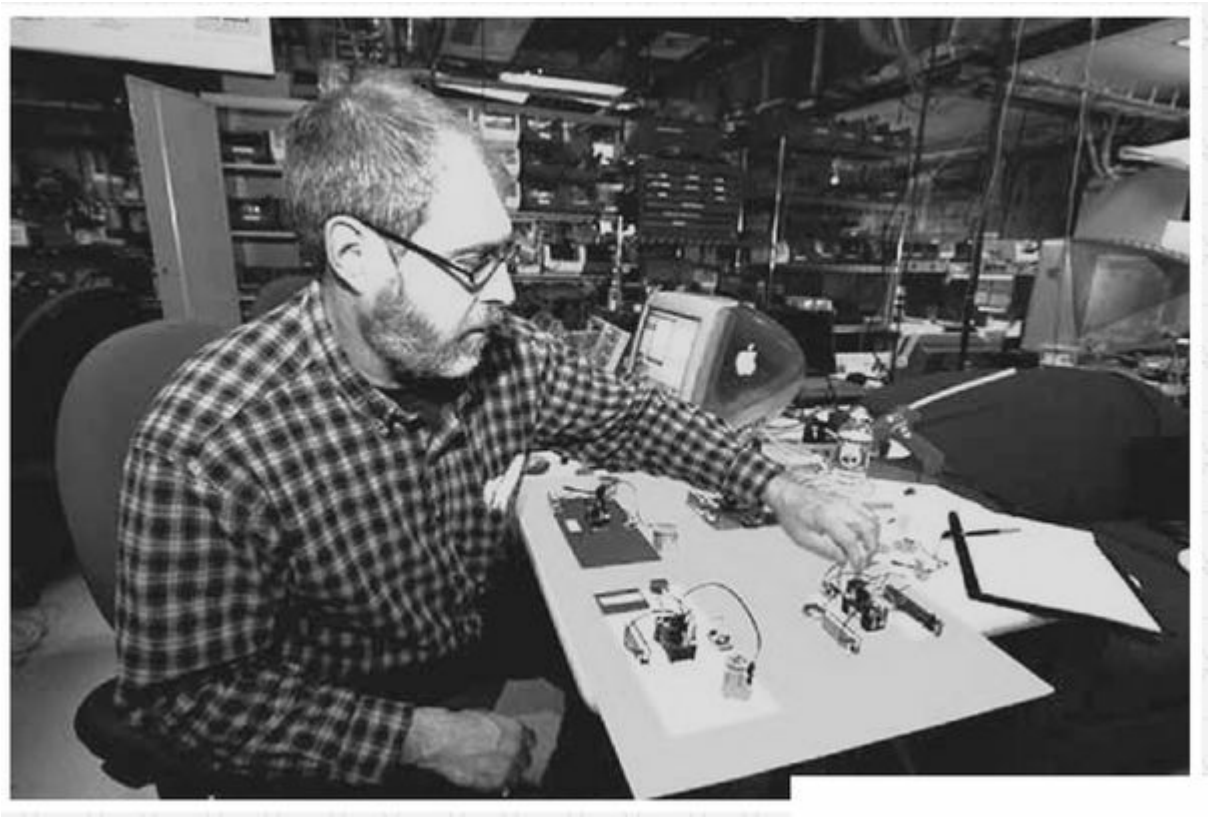
Ken

The biggest surprise when LEGO started selling Mindstorms was who bought them. Of the adults leaving toy stores with kits, about half turned out to be getting it for themselves rather than for children. Engineers who might work as small cogs in big projects were attracted by the opportunity to build complete systems themselves. And, even more

surprising, they weren't doing this just for fun. I first came across this when I was visiting a colleague doing physics research at a lab that had recently won a Nobel prize; he was happily assembling LEGO components to put together a control system for his latest experiment.

Back at MIT the same thing was happening, as grown-ups adopted as serious tools technological toys developed for kids. One of them was Ken Paul, a process engineer from the United States Postal Service. He was based on campus to investigate applications of emerging research results for USPS operations, which were expected to include things like putting radio tags in envelopes to help better route the mail.

As Ken Paul went about his business he did something a bit like what the girls did when the first computing bricks came into the classroom, when they discreetly watched what the boys were doing and then started adding intelligence to the playhouses they were building. Kenny eyed the kids playing with the new generations of computing bricks that Mitch, Bakhtiar, and their colleagues were developing, and he thought that what they were doing looked like more fun than what he was doing. More seriously, he wondered if they could be used to help model billion-dollar infrastructural decisions.



Serious work

The USPS operates on such a large scale that in its acquisition and operations budget, there's a dangerous gap between funding an exploratory paper study and committing to a large-scale deployment. It's not possible to build a football-field-sized mail-sorting facility as a prototype to evaluate and optimize the design, so there's a leap of faith to go from a design to actually building the facility. Unless, of course, the football field can fit on a tabletop. Ken started working with Bakhtiar Mikhak and Tim Gorton at MIT to make LEGO-scale models of these big installations, which looked like toys but operated according to the rules of their real counterparts.

He was hooked. As he progressed, with some trepidation we began planning a visit to Washington, DC, to present the project to USPS management. Ken worried that the project might appear to be pretty frivolous when he was seen playing with toys.

We trooped down to Washington, arrived at the executive conference room, and set up his system. The management team then filed in, looking suitably serious. Eyebrows raised when they saw what looked more like a playroom. But as Kenny explained what he was doing, and then did a demonstration, their eyes widened. Managers who spent their days on the receiving end of thick reports asked whether even they could use a tabletop prototyping system to try out what-if scenarios themselves instead of paying high-priced consultants to do the analyses.

The importance of Ken's demonstration went far beyond the particular mail-handling application he had chosen. Just as spreadsheets became a killer application for PCs because they allowed managers to make models of their financial flows, the computing bricks allowed the USPS managers to model physical flows of materials and information. Business executives as well as kids like hands-on interfaces, immediate feedback on their actions, and the ability to work together

in groups to solve problems. There's very little difference in the technology for serious work and serious play.

Amy

Amy Sun was a prized defense engineer, doing critical work for key government programs on earth and in space. Outside of her day job she had a consuming interest in engaging kids in science and technology, ranging from helping with school outreach programs to building battling robots. I met her in 2002 when she joined my colleagues Professor Ike Chuang, CBA's program manager Sherry Lassiter, and MIT undergrad Caroline McEnnis, in setting up our first field fab lab, in India at Kalbag's Vigyan Ashram (a rural science school described in "Making Sense").

Before going to India, Amy stopped by MIT to help get everything ready, and in the best tradition of incoming MIT students, she promptly took over the lab. She quickly mastered the larger on-campus versions of the tabletop rapid-prototyping tools going into the field, using them to make parts needed for the trip, and much more. Her decision after the trip to become a grad student at MIT followed more as a statement than as a question; she (correctly) assumed that MIT's application formalities were just a detail, given the obvious match with her abilities and interests.

After a delay while she handed off her projects, she interrupted her previously scheduled life to come to MIT as a student. Along with her research on campus, which entailed shrinking LEGO bricks down to a millionth of a meter in order to assemble microscopic three-dimensional structures, Amy was soon back in the field helping start fab labs in inner-city Boston in 2003 at Mel King's South End Technology Center, and in 2004 in Ghana at the Takoradi Technical Institute.

She created quite a stir when she arrived in Africa; a typical comment that came back while she was there was, "The lady is amazing she got every stuff in her!" Even better, she was such a strong role model as an engineer that boys using the fab lab asked her questions that had perhaps never been asked before: "Is MIT a girls' school? Will they let boys in?"

Amy started off fab lab users in Africa by having them use the laser cutter to make puzzles that they had to assemble into geometrical shapes. This was intended to be a warm-up lesson for the students, but it turned into an even greater lesson for the teachers: The street kids coming into the lab were able to solve the puzzles much more quickly than the older students or adults there.

Ironically, while this was going on one of the older teachers in the school asked why the high-tech tools in the fab lab were being wasted on young children. It was an eye-opening moment for him when he saw what those kids could do. Not too long afterwards, a local reverend admonished the community to go home and start fasting and praying because "the Bible tells us that we should be telling the children to come unto and instead we have been pushing our children away." He was struck by finding that a resource as valuable as the fab lab could be shared with kids rather than protected from them. That's a lesson that kids of all ages in the lab did understand. A single seat might be occupied by ten bottoms, as young and old, boy and girl, piled on to share a tool, instinctively collaborating and teaching one another.

Teaching technology

Problem solvers

As work in the fab lab progressed from teaching tutorials to working on real-world applications, it ran into a significant

limitation. One of the highest priorities to quickly emerge in the Ghana fab lab was solar energy, seeking to develop machines that could directly harness the abundant power of concentrated sunlight without the cost and inefficiency of converting it to electricity first. This activity required making solar collectors that were not only bigger than the cutting tools in the fab lab but needed to be bigger than the lab itself. But how can a machine make something bigger than itself?

An answer can be found all the way back in Seymour Papert's turtle. The fab lab could make a mobile computer-controlled car that could drive over the material to be used, trailing a pen to plot precise shapes that could then be cut out with simple hand tools. The original turtles provided graphical feedback long before computer graphics were up to the job; this new kind of turtle could do the same for fabrication. A project soon followed to develop a turtle design that could be produced in the fab lab.

This solution to the practical need to plot large structures represents the realization of one of Seymour Papert's original dreams. As exciting as bringing computers and controllers into the classroom once was, he describes the inability of the kids back then to invent as well as use technology as a "thorn in our flesh." Unlike what's possible with the materials in any well-equipped arts-and-crafts area, the turtle provided output from the computer but the physical form of the turtle itself couldn't be changed. Its fixed form put a severe bound on the shape of kids' ideas.

The possibility now of making a turtle in a fab lab is important as an application, allowing the production of structures bigger than the lab. The implications range all the way up to global energy economics in fabricating large-scale solar collectors. But a do-it-yourself turtle is even more important as a process, breaking down the barrier between using and creating technological tools. Like real turtles, one starting design can evolve into many subspecies. The inspiration for such an invention can be play, or work, and, best of all, because it can be done by an individual rather than justified by an organization it's not necessary to try to tell the difference.