

# SmartTiles: Mobility and Wireless Programmability in Children’s Construction and Crafts

Nwanua Elumeze and Michael Eisenberg  
Department of Computer Science  
University of Colorado  
Boulder, CO USA 80309-0430

## Abstract.

This paper presents a working prototype of a mobile, programmable set of construction kit elements for children. *SmartTiles* are small, lightweight, independently programmable tile objects that can be combined to cover various sorts of planar surfaces; each touch-sensitive tile contains its own computer and LED, and communicates with its neighboring tiles when placed on an appropriate background material. Collectively, the tiles enact user-customizable cellular automaton programs and thus display complex and fascinating dynamical patterns of light. In this paper, we discuss the implementation of SmartTiles and explore their potential use as an instance of mobile computation for children. We also discuss the way in which the tiles can be programmed wirelessly via a PDA interface, and discuss the implications of this sort of programming for educational computing more generally.

## Keywords

Computers and education; SmartTiles; mobile computing; computationally-enhanced construction kits; wireless programming.

## 1. Introduction: Computationally-Enhanced Construction Kits for Children

The term “mobile computing” can take on a variety of meanings in discussions of education. Often, the term is used to describe a style of computing in which traditional desktop applications are re-implemented with a simpler or sparser interface, appropriate for use in handheld computers. The utility of this sort of “mobile computing” is undeniable, but it largely represents a translation of one form of educational technology—a form devised with a desktop machine in mind—to a more portable format.

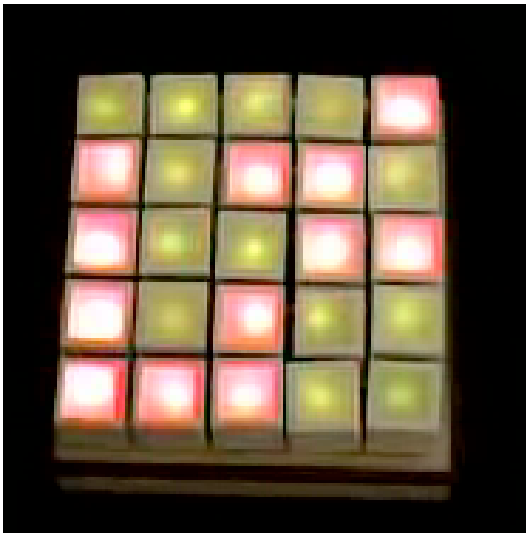
This paper describes a type of “mobile computing” in which the affordances of computation are woven into a novel type of construction kit for children. The basic idea behind this work is that the educational strengths of construction kits—their use as a creative, flexible, content-rich medium for children’s design—can be enhanced by the incorporation of computational control and programming. Construction kit elements can incorporate embedded computers, and can thus be capable of interesting behaviors; and when such elements are combined into larger, composite constructions, the various computational pieces can interact in fascinating and complex ways.



**Figure 1:** Four SmartTiles (at about 1 cubic inch, each is small enough to fit easily in a child’s hand). The tile at upper right is a cylindrical shape, while the other three are in the “standard” cubical form.

The system described in this paper, *SmartTiles*, is a working prototype of just such a computational construction kit. SmartTiles is a set of small “tile-like” pieces that can be used to cover surfaces: four such pieces (three cube-shaped pieces and a cylindrical piece) are shown in Figure 1. We will go into greater detail about the implementation of SmartTiles shortly, but a distilled description is

worthwhile here. Briefly, sets of SmartTiles are placed in a background fabric that provides the power and communication pattern for the set of tiles; in a typical setting, with a large fabric platform placed (e.g.) on the wall of a classroom, hundreds of tiles might be placed in a planar array. Each SmartTile contains a microcontroller that dictates the rules by which it communicates with its neighboring tiles; typically, each tile also includes an LED light and is touch-sensitive (other variations are possible). When placed together into an array, such as the small five-by-five version shown in Figure 2, the overall set of SmartTiles may be used to enact programs typical of cellular automata—i.e., collections of (individually simple) SmartTile programs may give rise to (collectively complex) emergent programs that are manifested over larger surfaces. A video of a five-by-five array of SmartTiles in action can be seen by following a link from the website: <http://l3d.cs.colorado.edu/~ctg/projects/smarttiles/tilesindex.html>. This video also illustrates the manner in which a tile's state may be altered by touch, and the way in which tiles may be removed from (and replaced into) the background fabric.



**Figure 2:** A five-by-five array of cubical SmartTiles, showing a pattern of red and green lights from the LEDs inside the tiles. A video of the tiles in action may be seen at the website mentioned in the text.

SmartTiles represent a striking but non-traditional variety of “mobile computing”—a variety whose origins are more plainly located in the realm of children’s construction and crafts than in desktop

computing applications. Each SmartTile is a stand-alone, lightweight “computational nugget” that can be removed from its background material, carried over to a desktop machine, and reprogrammed; once a new program is provided for the tile, the piece may then be replaced into the background material, and its behavior will be altered in accordance with its new instructions. Alternatively, the tiles need not even be removed from the background material in order to be reprogrammed: it is possible to endow an individual tile with new behaviors by programming the tile wirelessly from a handheld computer. Thus, SmartTiles point the way not only to a new type of computationally-enhanced set of construction pieces, but toward a flexible type of “in-place” reprogramming of construction kit pieces as well. As a working prototype, the SmartTiles system thus highlights a variety of fascinating issues surrounding the nature of mobile and wireless computing in the design of children’s artifacts.

The remainder of this paper is devoted to a more detailed description of SmartTiles, as viewed along several distinct dimensions. The second section focuses on the technical implementation of the tiles and the background fabric in which they are placed: in essence, this section views the tiles along the “engineering” dimension. The third section focuses on the ways in which tiles may be customized or programmed, either directly at a desktop machine or “in place” via wireless communication. This section begins with a short explanation of cellular automata (the genre of computing modeled by SmartTiles). The fourth section discusses the role that we eventually see SmartTiles playing in an educational setting (pilot tests with children are planned to begin later this year), and discusses the ways in which computational construction kits represent a blending of novel technology with some of the most powerful traditions in children’s craft activities. Finally, we discuss SmartTiles in the context of related work, and we describe our ongoing work and potential future directions for continued research.

## 2. SmartTiles: a Technical Description

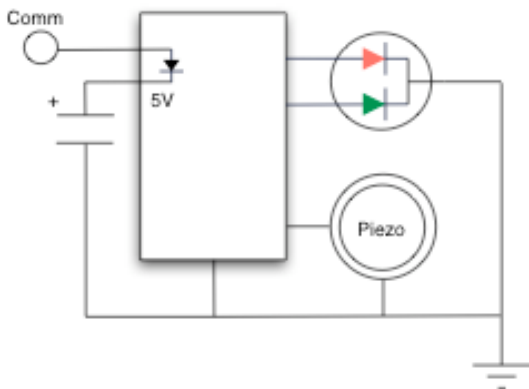
### 2.1 Overall Design Goals

The design of the SmartTile system has been conducted with an eye toward a particular set of design goals. These goals are (in our view) representative of the larger genre of

computationally-enhanced construction kits. Briefly, we have created SmartTiles to be:

- *programmable* (and re-programmable) by their users
- *interactive* (so that users may affect—for instance, via touch—the behaviors of constructions in action)
- *robust*, for repeated use by children
- *low-maintenance*
- *simple in design*, requiring relatively modest components (including computational components)

The SmartTiles system consists of 3 identifiable components: the tiles themselves (i.e., the individual pieces of the sort shown in Figure 1); a base (which we will generally refer to as the “background fabric”) that provides the infrastructure for communications and power; and a wireless conduit. This last element will be described in the following section: essentially it provides the communication infrastructure and protocol that enables children to wirelessly “beam” new programs to subsets of tiles or to an entire array of the tiles simultaneously.



**Figure 3:** Each smart tile contains a 5 MIPS mixed-signal micro-controller, a tri-color LED, and a piezoelectric disk. Voltage on the communication line is rectified with the external 100uF capacitor and internal diode to provide stable chip power.

## 2.2 Tile and Background Fabric Design

A schematic of an individual tile is shown in Figure 3: this is a “standard” tile, consisting of a

microcontroller, tri-color LED output, and piezoelectric disk to make the tile touch-sensitive. Variations on this design are possible: for instance, rather than (or in addition to) an LED output, one could create a “sound tile” capable of playing (e.g.) a tone or musical motif instead of merely shining light of a particular color.

When connected to the fabric that provides common power and communication connections, the tiles become participants in a global network topology, where they all share a single communication and power line. This approach reduces the number of connections a tile would have to make, which reduces pin-count and simplifies the hardware design, as suggested by the design goals noted earlier. Figure 3 illustrates the final circuitry of a smart tile where in fact only two pins are exposed to the outside world: *power/comm* and *ground*.

In effect, a set of tiles placed in the background fabric may be thought of as so many separate computers, each running a simple program at every tick of a synchronized clock signal. Often—in many typical scenarios—it will be the case that every tile is running the *same* program at each tick of the clock. This need not always be the case, however: each tile contains its own individually defined program, and conceivably every single tile could run a program that happens to be distinct from that of every other. Each tile also has its own internally-defined state variables; and a tile’s program can make use of the values of those variables for itself and for its immediate neighbors. Those readers familiar with the topic of cellular automaton programming will recognize the SmartTiles system as typical of this genre of programming; we will go into a bit more detail on this subject in the following section.

## 3. SmartTiles Programming

In this section, we discuss the techniques by which SmartTiles may be programmed. As a preface to that discussion—since SmartTiles, as noted, exemplify cellular automata—we begin with a brief explanation of that style of programming.

### 3.1 Cellular Automata and Their Programming

The subject of cellular automata is one with a venerable history, and extensive literature, in computer science; see [5, 9] for good introductions to the subject. Essentially, cellular automata are

collections of very simple computers, connected in a regular geometric arrangement and running in synchronized fashion from a common clock signal. In a typical automaton, each cell occupies an individual square in a tiling of the plane, as suggested by the five-by-five array of Figure 2; while Figure 2 depicts a small number of cells, a more typical automaton in the literature will have a larger size, often numbering in the hundreds. (Although the square-tiling version is most common, other geometric arrangements are possible—some automata, for instance, are connected according to a tiling of the plane by hexagons.)

At each successive tick of the shared clock, each cell in the automaton runs a (presumably very simple) program that updates the state of that cell based on a function of its own current state and that of its immediate neighbors. Again, many variations are possible here, but in the “standard” square tiling situation, each cell’s neighbors consist of its eight surrounding square cells. (Cells at the edge of a finite array may employ “wraparound” to determine a full set of eight neighbors; for example, in the five-by-five array of Figure 2, the rightmost column is considered to be adjacent to the leftmost column, and the bottom row to the top row.)

An example may help to make this description more concrete. Here, then, is the program associated with each cell in a popular automaton that implements the “Game of Life”[2, 5]. In this system, each cell can be in one of two states—“alive” or “dead”. (These may be thought of as “lit” or “unlit”, respectively, for cells equipped with an LED output.) At each tick of the clock, for time  $T$ , each cell examines its own state and that of its eight immediate neighbors corresponding to the time  $T-1$ . Each cell then runs the following program, as expressed in pseudocode:

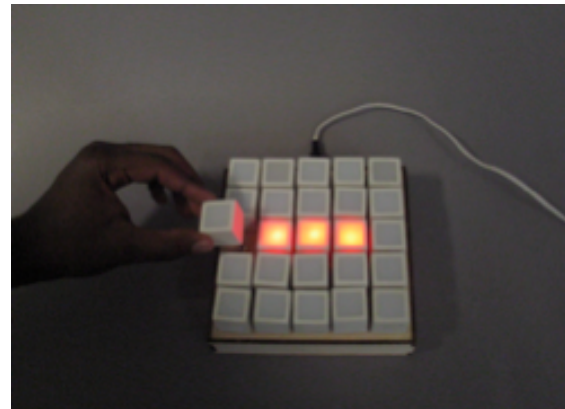
```

IF my-state = “alive” at time T-1:
    IF the number of live neighbors = 2 or 3
        THEN my-state = “alive” at T
        ELSE my-state = “dead” at T
    IF the number of live neighbors = 3
        THEN my-state = “alive” at T
        ELSE my-state = “dead” at T

```

The Game of Life, created by the mathematician John Conway and popularized by Martin Gardner in his *Scientific American* columns in the early 1970s, became an enormously popular subject of recreational computing.[2] In this system, very

simple starting configurations of “live” cells on a grid can give rise to remarkably fascinating dynamical patterns over time: sets of live cells that appear to travel across the plane (“gliders”), sets of cells that shift back and forth between finite patterns (“oscillators”), apparently chaotic and unpredictable patterns, and so forth. (Again, [5] is an invaluable nontechnical reference here.) The crucial point of mathematical and computational interest is that large numbers of extremely simple computational components—the cells—when connected into regular patterns of communication, are capable of astonishingly complex overall behaviors. The Game of Life is only one of a multitude of fascinating automaton systems along these lines; [9] is a remarkable reference book that discusses a range of automaton systems modeling important ideas from fields such as physics and chemistry.

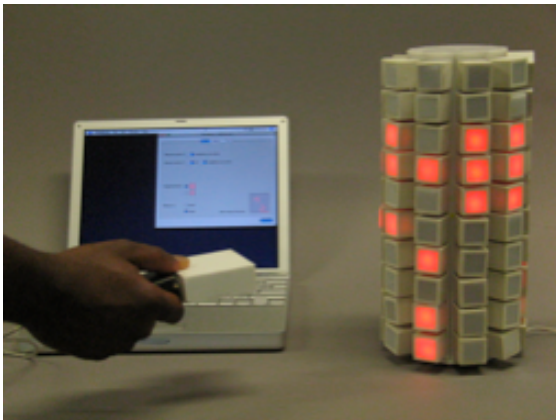


**Figure 4:** Removing a tile from an array in preparation for reprogramming it at a desktop machine.

### 3.2 Programming SmartTiles to Enact Cellular Automaton Rules

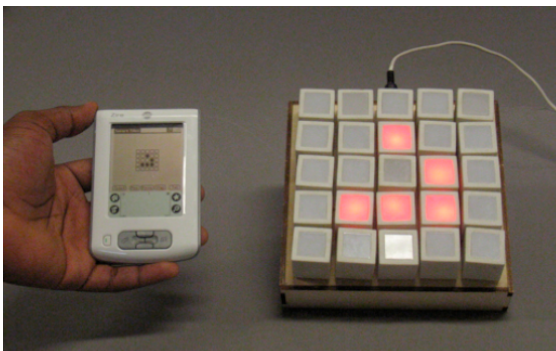
The programming system that we have developed for SmartTiles allows for the development of a potentially huge number of distinct cellular automata programs. There are in fact several ways of interacting with SmartTiles. First, an *individual* tile may be removed from a set of tiles, brought over to a desktop computer interface, and given its own particular program. (This is *individual tile programming*.) For example, we may want to have a large collection of tiles running the Game of Life, but to designate one or two specific tiles as “source” cells that are always alive throughout the running of the overall system: in this case, the “source” program would simply be to remain alive at every tick of the clock, regardless of the state of

one's neighbors. Figure 4 shows a single tile being removed from an array of tiles in order to be reprogrammed at a desktop machine.

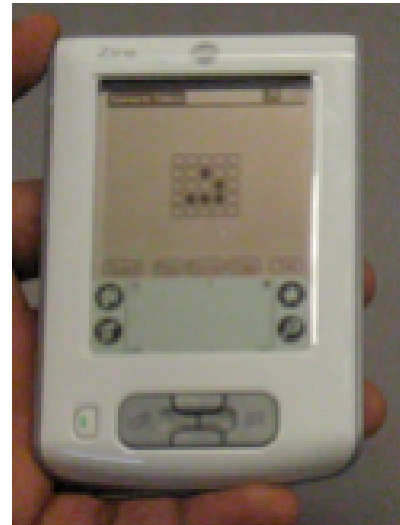


**Figure 5:** A group of SmartTiles in a cylindrical arrangement being programmed as a group via wireless connection from a desktop computer. (As an aside, the cylindrical pattern of tiles will be discussed in Section 4 to follow.)

The second means of programming SmartTiles is also through a desktop computer, via a wireless interface that can communicate with all the cells through the medium of the background fabric. In a typical case, the user would wish to give all the cells, or some designated subset of them, a particular program. (This is *group tile programming*; Figure 5 shows an example of this sort of interaction.) Here, the tiles themselves need not be removed from the background fabric in order to be reprogrammed.



**Figure 6:** An arrangement of SmartTiles being dictated via a handheld computer.



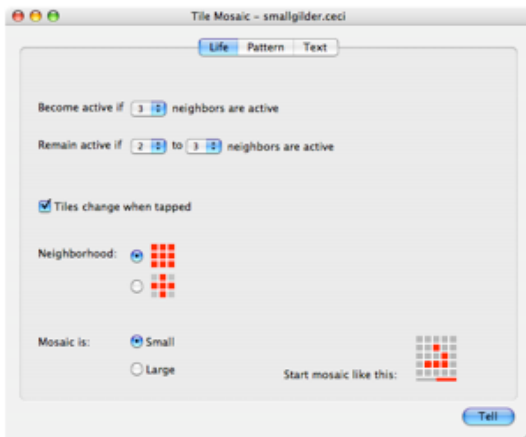
**Figure 7:** A “close-up” of the handheld computer interface shown in Figure 6. Here, the user is initializing the set of lit (or “alive”) cells to the configuration on the screen of the handheld device.

The third means of interacting with SmartTiles employs a handheld computer that can be used to (e.g.) reprogram tiles, to change the state of individual tiles, and to speed up, slow down, pause, or restart a simulation. (This is *handheld wireless interaction* with a SmartTiles system.) This last mode of interacting with SmartTiles, because it does not make use of the full desktop programming interface, is less powerful than the other two means; but it enables users to reprogram and interact with running automaton programs in the absence of a desktop computer. Figure 6 shows a handheld device being used to set an array of cells (those familiar with the Game of Life will recognize this set of five live cells to be the well-known “glider” configuration). Figure 7 shows a close-up of the handheld interface, depicting the configuration that the user is sending to the array of cells in Figure 6.

### 3.3 Programming the Tiles: “Life” Interface and Language

In our current implementation of SmartTiles, there are two major ways of programming the tiles from a desktop machine. One method employs a menu-driven “Game-of-Life-style” programming system; this is appropriate for endowing tiles with programs similar in structure to that of the Game of Life. In particular, within this interface, tiles are assumed to have two states (“live/active” or “dead”); a live cell remains alive if surrounded by  $N$  live neighbors, where  $N$  is chosen by the user; and a dead cell

becomes alive at the next time step if surrounded by anywhere from  $K$  to  $M$  live neighbors, where  $K$  and  $M$  are chosen by the user. (Thus, for the game of life,  $K$  has a value of 2, and  $M$  and  $N$  both have a value of 3.) Although very simple to use, the range of automaton algorithms expressible through this interface are distinctly limited. A view of the interface can be seen in Figure 8.



**Figure 8:** A view of the menu-driven “Game-of-Life-style” interface for programming SmartTiles.

The second and more powerful (but also more challenging) way of creating SmartTiles programs makes use of a textual language, similar in syntactic structure to the Logo language and expressly geared toward writing cellular automaton programs. A full description of this language is beyond the scope of this paper (and in any event, the language is still a work-in-progress); but to provide an example, the central text in writing a Game of Life program for SmartTiles appears as follows in the tile language:

```

LOOP
[WAITSYNC 1
  IFELSE RED
    [IF RED-NEIGHBORS < 2 ||
      RED-NEIGHBORS > 3 [TURN OFF]]
    [IF RED-NEIGHBORS == 3 [TURN RED]]
]

```

The tile language is currently rich enough to express a tremendous range of important ideas in cellular automaton programming, such as: three- or four-state automata (with red, green, and amber lights representing state values, as well as “off”); probabilistic rules (e.g., a cell might have a 10 percent chance of changing state given certain surrounding conditions); rules involving simple

arithmetic (e.g., a cell might change its state if and only if there are exactly three more red than green neighbors); and so forth.

In the case of handheld wireless interaction, the only mode of reprogramming tiles is through the textual language (that is, there is no menu-driven interface analogous to that on the desktop machine). Figure 9 shows a screen view of a handheld device in the course of sending a new program to an array of SmartTiles.



**Figure 9:** A screen view of a handheld device being used to reprogram, textually, a set of SmartTiles.

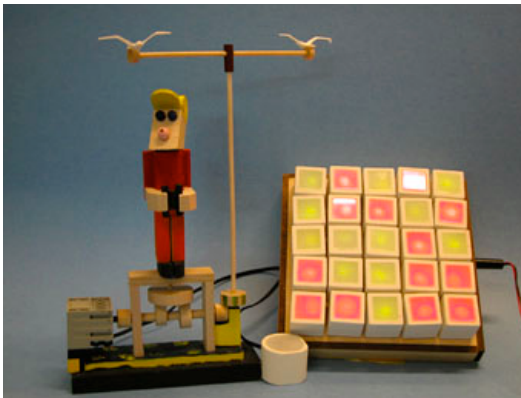
#### 4. SmartTiles as an Educational Artifact

The previous sections have focused on what SmartTiles are; how they are implemented; and how programs can be written for them. In this section, we turn to an exploration of SmartTiles in their role as an educational artifact.

To return to an earlier theme of this paper: in many respects, it is possible to think of SmartTiles as representative of a genre of computational construction kits. Here, the “pieces” are bits of computation (tiles) that can be removed from an array, reprogrammed in a customized way, and placed back into their original setting. Unlike traditional construction kits, then—whose subject matter is (e.g.) architecture or mechanics—the primary subject matter of SmartTiles may be viewed as that suggested by cellular automata. That is, SmartTiles highlight issues of dynamical systems, emergent behavior, collective

organization, and so forth. (See [7] for a compelling discussion of the role of these themes in education.)

There are many other themes that are worth discussing in this light, however, beyond the mathematical or procedural content of a SmartTiles program. For example, consider the fact that SmartTiles (unlike, say, cellular automata as portrayed on a computer screen) can be combined in large numbers to make a mural-sized display (e.g., on a bedroom wall). Although our current prototype includes only a relatively small number of SmartTiles, there is nothing in our design that precludes the possibility of very large collections of tiles placed over substantial areas in a classroom, child's room, office building lobby, and so forth. We could imagine a child's room in which a collection of SmartTiles is spread in gleeful proliferation over walls, ceiling, shelf, and desktop.



**Figure 10:** One of the tiles in this SmartTiles array is connected to a motor that activates the wooden birdwatcher.

In effect, then, SmartTiles point toward a style of “ambient educational computing” in which constructions and simulations are spread throughout an entire setting—a room, or hallway, or garage. Rather than confine educational simulations to a desktop screen, they can instead be diffused in unpredictable and creative ways throughout a child's physical space. Moreover—and also quite important in this context—one should not think of SmartTiles as limited in their behavior to simply flashing lights on and off: that is, LEDs need not be the only “output” signal of SmartTiles. As mentioned in Section 2 earlier, it is possible to design a “musical SmartTile” that instead of light uses sound as its primary mode of output. (We have created a prototype tile of just this sort.) Another possibility is to have SmartTiles send an output

signal to a device (such as an electric motor) rather than to an LED. In this way, a tile might not flash a light, but might instead be used to turn on and off other devices throughout a child's room. Figure 10 shows a (very tiny, but still suggestive) example of this idea in action. Here, one of the SmartTiles in an array is connected to a motor that activates a commercially-available wooden mechanical toy. (Thus, in this system, every time the tile is active, the mechanical birds revolve about and the bird watcher turns his head.)

Additionally, there is no particular reason to confine SmartTiles to flat, planar arrays. More creatively, these tiles could be thought of as “surface coverings” in the spirit of mosaic tiles. An example (again, relatively simple) is suggested by the cylindrical pattern of tiles shown in Figure 5 earlier. Here, the background fabric on which the tiles have been placed happens to be a cylinder instead of a flat plane; the resulting set of tiles might be used (e.g.) as a hanging decoration from a ceiling.

In effect, the discussion of the past several paragraphs is intended to suggest a collection of novel and interesting directions for educational computing in general. Rather than think of “educational computing” as limited to a computer screen, it can instead be viewed as subject matter deeply interwoven with environmental design. “Educational computing” can and should take place at the scale of the room, the house, or even the neighborhood, rather than simply in a tiny square display area on a desktop. SmartTiles represent one (still early) foray into this new territory of educational computing: the tiles eventually might cover large and varied surfaces in all sorts of distinct places, endowing those surfaces with interesting procedural behaviors as reflected in light, sound, and motion.

Because SmartTiles are small, portable, and (one can plausibly argue) relatively inexpensive, they also move educational computing into still other realms historically associated with children's culture. In large numbers, it would be conceivable that tiles could take on the status of “collectibles” in groups of children—collectibles that are individually customizable with their own particular behaviors. Tiles might be (e.g.) traded or given as gifts in ways that have heretofore largely eluded computational artifacts (it is the rare screen-based simulation that is given as a gift, or traded for another simulation).

Finally, SmartTiles suggest new directions for exploration in children's programming. Historically, there have been fierce debates about the means, and even the utility, of providing programming media for children or "non-programmers" more generally. (See [1], [4] for cogent discussions of such issues.) We believe that much of this debate has been saddled by relatively traditional ideas of what sorts of programs users (or children) might need to write. In the case of SmartTiles—consistent with the field of cellular automata more generally—the actual programs that students are assumed to write are extremely short and simple. (It is a very rare cellular automaton program that takes more than half a page to express.) In other words, by designing programmable media for children in which the programming element is powerful but *distributed*, many of the classical arguments about children's programming—that it is too dauntingly complex, that it is time-consuming, that it is dull—carry far less intuitive force.

## 5. Related, Ongoing, and Future Work

Our own work in computational construction kits has been substantially influenced by the work of various other research groups, including those of Resnick [8] and Ishii [6] at the MIT Media Lab, and the work of Kitamura and colleagues on the ActiveCube project [3]. All these research groups are likewise developing means of incorporating computation into construction media for children, though there is a variety of stances taken on (e.g.) the issue of end-user programmability. (In this respect, we are closest to the approach of Resnick's group, focusing as they do on creating artifacts that can be programmed by their student users.)

In the near term, we intend to begin a series of pilot tests of SmartTiles with children at a variety of ages (focusing primarily on students of approximately middle school age). To date, the relatively small number of SmartTiles has precluded our ability to work with children on meaningful projects; but as the number of tiles expands, it becomes increasingly feasible to test the use of the tiles under realistic conditions (i.e., in the context of interesting cellular automaton simulations). Our first pilot tests are planned for later this year. In the longer term, our goal is to develop a wider range of tiles (e.g., musical and mechanical tiles of the sort described earlier), and to investigate the ways in

which children program the tiles through both handheld and desktop computers. In sum, we believe that SmartTiles will yield a tremendous variety of research and design questions; collectively, these should further the development of mobile, ambient, and end-user-programmable computational artifacts for children.

**Acknowledgments.** Thanks to Mark Gross, Michael Mills, Mitchell Nathan, Andee Rubin, Mitchel Resnick, Gerhard Fischer, Roy Pea, and Carol Strohecker for their conversation. This work was funded in part by the National Science Foundation (awards no. EIA-0326054, and REC0125363).

## References

- [1] DiSessa, A. [2000] *Changing Minds: Computers, Learning, and Literacy*. Cambridge, MA: MIT Press
- [2] Gardner, M. [1985] *Wheels, Life, and other Mathematical Amusements*. New York: W. H. Freeman.
- [3] Kitamura, Y. et al. [2001] Real-time 3D Interaction with ActiveCube. In *Proceedings of CHI 2001* (Extended Abstracts), Seattle, WA, pp. 355-356.
- [4] Nardi, B. [1993] *A Small Matter of Programming*. Cambridge, MA: MIT Press.
- [5] Poundstone, W. [1985] *The Recursive Universe*. New York: William Morrow.
- [6] Raffle, H.; Parkes, A.; and Ishii, H. [2004] Topobo: a Constructive Assembly System with Kinetic Memory. In *Proceedings of CHI 2004*, 647-654.
- [7] Resnick, M. [1994] *Turtles, Termites, and Traffic Jams*. Cambridge, MA: MIT Press.
- [8] Resnick, M., Martin, F., Berg, R., Borovoy, R., Colella, V., Kramer, K., and Silverman, B. (1998). Digital Manipulatives. In *Proceedings of CHI*.
- [9] Toffoli, T. and Margolus, N. [1987] *Cellular Automata Machines*. Cambridge, MA: MIT Press.