

Function Composition in Physical Chaining Applications

Edward De Guzman, Gary Hsieh

Department of Electrical Engineering and Computer Science

360 Soda Hall #1776

University of California, Berkeley, CA 94720, USA

Tel: 1-510-643-3125

Email: {edwardd, garyh}@eecs.berkeley.edu

ABSTRACT

We introduce the notion of function composition, an interaction technique employed in several graphical user interfaces (GUIs), for use in physical chaining applications. Allowing the user to perform function composition in systems utilizing tangible user interfaces (TUIs) for input allows for more efficient use of the interactive workspace and the physical icons. In addition, it allows the user to customize the mapping of a physical icon to a function through end-user physical macro programming. In this paper we discuss the design of function composition in physical applications, as well as lessons learned from implementing function composition in one sample application, an image manipulation application.

KEYWORDS: tangible user interfaces, physical programming, physical chaining

INTRODUCTION

Physical chaining applications are a subset of a larger class of systems which use tangible user interfaces [5]. Such interfaces attempt to bring computing off of the desktop and into a user's environment by coupling digital information to everyday physical objects and surfaces. This is one approach to making the act of computing ubiquitous and invisible.

In physical chaining applications, a group of physical objects representing digital data, functions or commands serving as input to a system, or arguments to these commands are placed on a workspace in a certain configuration in order to compute a result or display information (see Figure 1). The use of physical objects as physical icons (phicons) that can be arranged in a variety of ways affords their use as building blocks. The use of

Chain of phicons

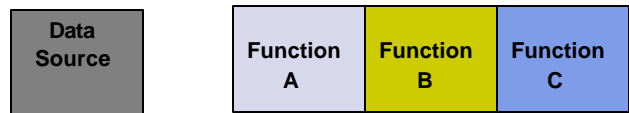


Figure 1: In this model of a physical chaining application, the data source would be modified by function A, followed by function B and then function C. The user specifies which functions and the order to use them by building this chain with physical icons.

building blocks as a metaphor for direct manipulation of digital data is the underlying motivation for physical chaining applications as a novel approach to computing. With a set of building blocks at his/her disposal, a variety of users (both young and old) can use them to quickly construct physical structures and manipulate already existing structures.

Through two-handed interaction, users can quickly see the effects of replacing one block with another. With the quick feedback received by the user, using physical building blocks promotes exploration of how the blocks interact with each other and the effect of placing blocks in various positions in the overall structure. Allowing users of physical chaining applications to interact with phicons as they would with building blocks gives users the opportunity to use their two hands to explore the possible ways of manipulating digital information. In addition, users can explore the relationships and resulting effects from different arrangement of these phicons. Physical chaining applications also provide a form of error-checking by constricting the user to build structures corresponding to inputs that can be appropriately handled by the system. Furthermore, an optimal design of the phicons could take advantage of the human understanding of basic concepts such as balance, gravity, and spatial visualization to prevent the user from providing nonsensical input to the system. For example, to prevent a user from performing function A after function B, one can design the phicons for these

functions so that phicon B cannot be attached to the end of phicon A.

Systems relying on GUIs for input also allow users to manipulate data and create complex information structures. However, widgets in most GUIs do not have affordances that are as visible in building blocks. Input is limited to mouse and keyboard which do have affordances as rich as building blocks. Another disadvantage of GUIs is the difficulty in initiating collaborative efforts with other users in the same room. Input, with its limitations to mouse and keyboard, generally accommodate only one person interacting directly with the system while collaborators crowd around the monitor and verbally give suggestions to the user at the system. On the other hand, physical chaining applications (and, in general, systems using TUIs on an interactive workspace) allow collaborators to gather around the interactive surface and provide each user the opportunity to interact with the system through the use of phicons.

While the existence of an interactive workspace and physical icons are advantages for physical chaining applications over similarly functioning applications on a desktop computer, it also provides new constraints. First, the interactive workspace serves as a limit to how long a chain can become. This, in turn, limits the space of inputs available to the user and the space of data the user can explore with a limited input set. Second, the number of phicons available serves as a limit to the space of input. Suppose a chain required the use of all phicons available to the user. Adding to this chain or replicating this chain would not be possible due to a lack of phicons.

In this paper, function composition is designed as a tangible user interface interaction technique meant to improve upon existing physical chaining applications by addressing both of these issues. By concatenating a chain of functions into one phicon, the likelihood of using all available space with one chain is reduced. This gives the user a larger input space to explore with chains involving more functions. In addition, by being able to map one phicon to multiple functions which would normally require multiple phicons, function composition affords the reuse of phicons once a chain has been saved.

The remainder of the paper is structured as follows: First, we describe related work in function composition and in physical chaining applications, noting the similarities and differences to our work in each. Next, we present the general design of function composition and results from implementing this interaction technique in an application. Finally, we discuss opportunities for future work in this research area.

RELATED WORK

Several groups have investigated the manipulation of digital information through a tangible user interface.

Triangles [3] is a physical/digital construction kit consisting of a set of identical, flat plastic triangles, each with a microprocessor inside and magnetic edge connectors. These connectors allow a user to build a chain by connecting triangles. Digital information is sent via electricity passing over the connectors. Applications developed for the Triangles system include non-linear storytelling, media configuration, and artistic expression applications.

An application belonging to the last category was *The Digital Veil* [2], an art installation for the 1997 *Ars Electronica* festival in Linz, Austria. In this application built from the Triangles system, users are able to control not only the output generated by specific Triangles, but are also able to assign and reassign meanings of groupings of Triangles during the course of an interaction. A user can connect up to four triangles together and attach the arrangement to an "input station". Upon pressing a button, the user can speak into a provided microphone. Their voice is sampled and mapped to the arrangement of Triangles connected to the input station.

Although the user is able to assign meaning to a group of Triangles, this meaning is lost when the arrangement is disassembled. Thus, the voice sample cannot be collapsed into one Triangle which would save space on the work surface. In addition, the designers of *The Digital Veil* imposed a hard limit to the maximum number of Triangles that can make up a configuration. Our implementation of function composition similarly allows users to assign meaning to phicons. However, this mapping is persistent and does not require the chain to be intact to preserve the mapping. Also, there is no limit to the number of phicons that make up a chain.

Several interfaces have treated the phicons as programming elements [4,6,8]. Systems using these interfaces have allowed users to connect physical objects to correspond to a physical representation of a function written in a particular programming language (e.g., Algol, Logo, Pascal). While function composition does allow for a form of end-user programming, the type of programming in our implementation is more akin to macro programming than programming in a structural language such as Pascal. The user can specify the order in which functions are applied, but has no access to neither control structures such as while or for-loops nor conditional statements.

Macro programming is supported in a variety of productivity software applications such as Microsoft Word and Excel. One analog to our work using GUIs is the support for automating tasks in Adobe Photoshop [1] where a series of Photoshop commands can be grouped into a single command known as an *action*. By clicking a Record button, commands are saved in sequential order to a specific action. Once it is saved and given a name, the user

can refer back to this action and manipulate it by adding/deleting commands, rearranging the order of execution, or adjusting function parameters.

One system closely related to our research is Michael Terry's TaskBlocks system [9]. Physical blocks called task blocks represent data transformations. These tangible objects can be joined to create a data manipulation pipeline which, in turn, would act upon a single data source. Input devices can be attached to individual task blocks, enabling users to directly adjust the parameters to a function. However, this system does not support any form of composition which would save the user space on the interactive surface. We hope to augment this model of computation involving a single data source and a chain of functions with support for function composition.

IMPLEMENTING FUNCTION COMPOSITION

In order to study this interaction technique, we have designed and implemented a physical chaining application that supports function composition. This application is built from the following components (see Figure 3):

- SMART Board: serves as the interactive workspace
- Cardboard paper phicons: units of the physical chain
- 17" color monitor: output display located next to the SMART Board
- Papier-Mâché: system for tracking locations of objects on the SMART Board and firing events when objects have been added, removed, or rearranged.

Details about each of these components are provided below.

SMART Board. A Rear Projection SMART Board 1602 (50 1/2" W x 76 1/2" H x 50 1/4"D) [7] is used to act as the interactive surface where chains are created, removed, and edited. The Rear Projection SMART Boards are large projection displays that are touch-screen sensitive. A Proxima projector points to a mirror which reflects a projected image onto the semi-translucent board.

Phicons. We have designed a set of phicons (see Figure 2) to serve as building blocks for the chain a user can build with our application. The square or rectangular-shaped phicons are made of cardboard. Square phicons represent "save blocks"--blocks that can store a chain of functions. Rectangular phicons represent function blocks which can perform one data manipulation function on the data source. To help users distinguish among the various blocks, we have labeled one side of each phicon with either a function name or a "Save" label. The length and width of the phicons are designed to be multiples of two centimeters. This maximizes the number of sizes of blocks that can be uniquely recognized when placed on the SMART Board.



Figure 2: Physical icons made out of cardboard for the imaging application.

Papier-Mâché. Papier-Mâché is a toolkit supporting tangible user-interface design currently being developed by Scott Klemmer as part of his dissertation research. It uses computer vision techniques to detect when objects are added, removed, and rearranged on the SMART Board, taking advantage of the Java Media Framework (JMF) and the Java Advanced Imaging (JAI) toolkit. A camera is placed inside the SMART Board cabinet pointing at the reflection mirror. When a block is placed on the board, its shadow will appear behind the semi-translucent board where the camera is pointing. Depending on the amount of light that passing through the SMART Board, the toolkit is able to sense when an object is placed or removed. Each frame of video is analyzed to detect the presence of a physical object on the SMART Board.

For our application, Papier-Mâché offers three types of events: add, update, and remove. Add events occur when a block is added onto the Board, update events occur when the orientation of the block changes, and remove events occur when a block is removed from the board. All three types of events give the programmer access to the position and size of the block being sensed.

Output Monitor. As the user provides input to the physical chaining application by placing blocks on the SMART Board, output is sent to a nearby display. In our application, we used a 17" CRT monitor with a resolution of 1024 x 768 pixels.

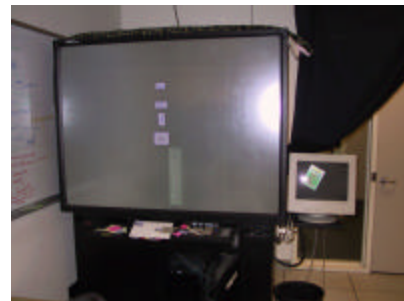


Figure 3: System components for physical chaining application, including the SMART Board, phicons, and output monitor.

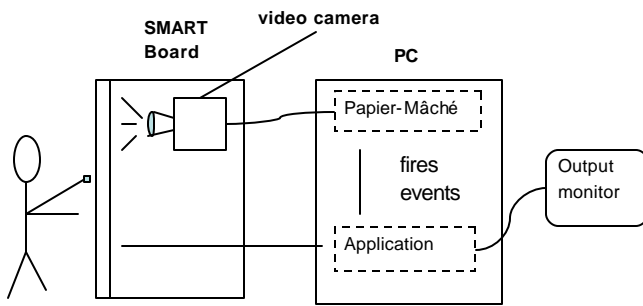


Figure 4: Event flow diagram for physical chaining application

Event Flow

The general program flow from user input to output on the nearby display occurs as follows (see Figure 4):

- User attaches blocks on the SMART Board
- Video camera captures input by recording activity on the SMART Board from the opposite side of the user.
- Papier-Mâché receives video from camera and analyzes each frame, firing events (add, update, or remove) to the software application.
- By analyzing the size and location of the event, the software application identifies the block and manipulates digital information accordingly
- Application sends output to a nearby display. If necessary, the application can also prompt the user for more specific input by projecting a GUI widget (e.g., a slider) on the SMART Board.

Imaging Application

Given this general architecture for a physical chaining application, we built a test application, focusing on image manipulation as the domain. In this application, blocks represent functions which could be applied to a scanned image. A chain of blocks would then represent executing each function on a single image. The output monitor would display the effects of each manipulation.

Screen Layout When the imaging application is started, a window outlining the workspace is projected on the SMART Board (see Figure 5). This space is partitioned into five regions:

- **Chain Area:** Users can create chains in one of four columns which make up the Chain Area by attaching a block in the appropriate column on the SMART Board.
- **Save Area:** Upon placing a save block in this area, the contents of the Chain Area located directly below this Save Area are saved in the save block.
- **Load Area:** This area is a portion of the rightmost column of the workspace where save blocks can be

placed. Upon sensing the presence of a save block, a graphical representation of the save block's contents is projected in the Edit Area directly below.

- **Edit Area:** The contents of a save block are projected in this area in the form of a vertical column of squares—one for each function (see Figure). Within this area, the user can rearrange the order of functions in the chain (using Java Swing's drag-and-drop technology), delete functions from a chain, modify the parameters of a function in the chain, or insert a new function in the chain via the Insert Area.
- **Insert Area:** Users can add a function to the end of the chain projected in the edit area by placing a block (either a function or save block) in this area.

Imaging Functions. The imaging application currently supports the following functions—rescale, rotate, flip (over a horizontal or vertical axis), and grayscale (convert to a black and white image). When a rescale or rotate function block is placed on the workspace, a slider is projected near the block (see Figure), allowing the user to provide a parameter for the function (e.g., scaling factor, number of degrees to rotate). Each of these functions are implemented using the Java Advanced Imaging (JAI) Package.

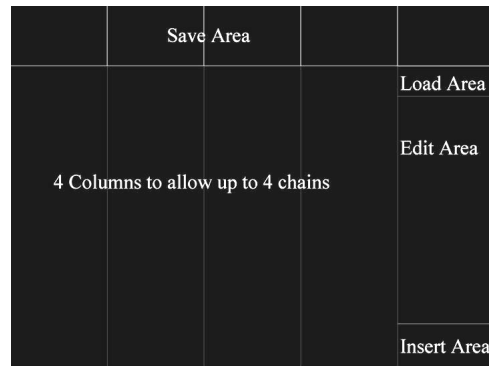


Figure 5: Screen layout for imaging application.

Block Recognition and Tracking. When the application receives an AddEvent, the block that caused this event to fire is recognized by calling the `getSize()` method of the event. This returns the number of edge pixels detected by the vision system. Given this metric for the size of the object, the block is identified by determining in which range the size falls. For example, a block with a size between 60 and 100 pixels is identified as a save block. If a block does not fall within a valid range, no action is taken.

If a block is successfully recognized by either a AddEvent or a RemoveEvent, a reference to it is added/removed in `blockList`, a Vector representing the list of all objects placed on the SMART Board.

Loading/Editing Images. Upon receiving an AddEvent in a Chain Area that was previously empty, a new copy of the

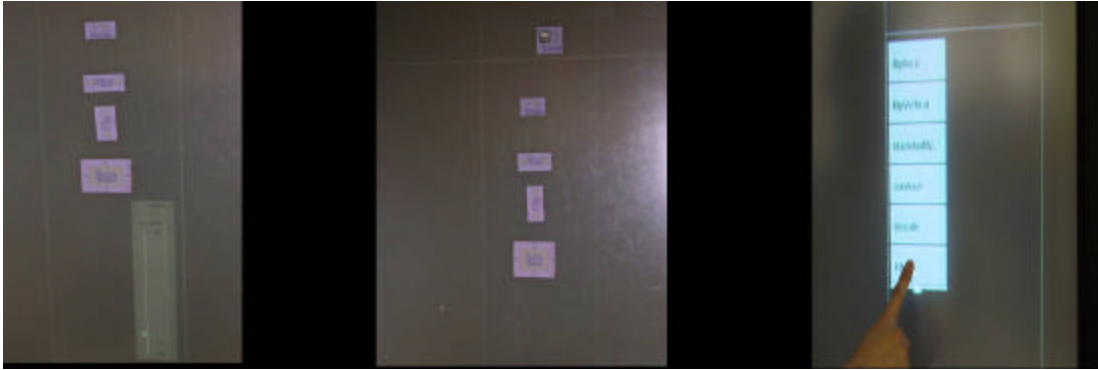


Figure 6: Demonstrations of various features of our imaging application are pictured above. Left: the application recognizes a rescale block and projects a slider, prompting the user for more input. Center: user performs function composition by saving a chain in a save block. Right: user manipulates a digital representation of a chain.

image is projected on the output monitor and the first function is applied. If two columns are used, the side screen is split into two halves; if four are used the screen is divided into quadrants. As the user adds more functions to an existing chain or creates new ones, the output monitor is updated accordingly in real-time. Similarly, when the application receives RemoveEvents and UpdateEvents, the chain and its corresponding image are modified.

Function Composition in the Imaging Application: A Scenario

Our support for function composition and possible uses of this interaction technique in physical chaining applications are demonstrated by the following key features of the imaging application (see Figure 6):

Creating Chains. Suppose a user wished to create a function chain of the functions “flip horizontal”, “rotate 30°”, and “grayscale”. This could be done by placing the blocks corresponding to each of these functions onto the SMART Board in one of the four available columns as well as in the order in which he/she would like the functions to be performed. When the user placed a rotate block on the SMART Board, a popup slider will appear next to the block prompting the user for the parameter (degrees to rotate in this example) to this function. The resulting image after performing these functions would be displayed in the output monitor.

Saving. If the user wished to repeat this series of functions on a different image, he/she can do so without rebuilding the chain by saving it in a save block, taking advantage of our support for function composition. To save a chain, the user must place a save block in the Save Area directly above the chain. Once the save block is recognized, the series of functions stored in the associated column is copied to the save block.

Editing. After filling several save blocks with chains of functions, suppose the user wished to change the argument of a rotate function from 30° to 60°. Now that function composition has been used to save chains, it is now

possible to modify chains without rebuilding them. Both function blocks (which require arguments) and save blocks can be edited. To edit a function block the user must tap on the block to modify. This will cause a popup to be projected on the SMART Board, prompting the user for the new value for the function argument.

Editing of save blocks can be accomplished in two ways. Like the editing of function blocks, editing can be initiated by tapping the block on the SMART Board. Alternatively, the user can move the save block to the Load Area. In either case, a digital version of the chain of functions stored in the save block will immediately appear in the Edit Area.

The Edit Area allows the user to edit a save block in various ways. To change the argument of one function, the user must tap on the digital block. The slider will then be displayed, allowing the user to specify a new value. In the same panel as the slider, a “delete” button will also be visible, allowing the user to delete that specific function from the chain.

To modify the order in which functions are executed within a chain, we allow the user to drag a digital block to a new position in the chain. If a user wishes to swap the position of two blocks, the user can click on the two digital images one after the other. This will cause a panel containing a “switch” button to pop up. Tapping on this button will swap the positions of the two functions in the chain.

Audio Feedback

As mentioned in the section “Block Recognition and Tracking”, the size of an addEvent must lie within a set range so that the block firing this event can be recognized. Due to the fluctuations in the size of events received from Papier-Mâché, it may require several addEvents before a block is recognized. In order to notify the user when an attempt to recognize a block has been successful, we provide audio feedback in the form of a small audio clip. A ringing tone is heard when a block is successfully recognized by the system. A slightly lower pitched tone is

played when a chain has been successfully saved in a save block. The notification tone heard when popup windows appear in Microsoft Internet Explorer is used in our application to alert the user when a side panel has been projected on the SMART Board (e.g., when the user must specify an argument to a function, when the user wishes to swap positions of two functions in edit mode). In contrast to the high notification sounds, a low pitched sound is played when the user places a block on the SMART Board or taps in an area where there is no physical or graphical block present.

PRELIMINARY USER FEEDBACK

Two Berkeley undergraduate students (one majoring in Electrical Engineering and Computer Science, the other in Civil Engineering) participated in an informal user study in order to obtain feedback on this first iteration of our physical chaining application. After being introduced to the goal of our project and the application we built, the two students were able to experiment with our application for approximately fifteen minutes each individually. During this session, the students were able to ask questions about how certain functions work. One designer was available to answer these questions, as well as guide the students in using the key features of our application by suggesting certain tasks to perform. Afterwards, the students were asked to provide comments about their experience in the same informal interview session.

Both students mentioned how the system seemed to be “slow”. This comment is probably in response to the several times when both students had placed a block on the SMART Board and had to wait for approximately ten seconds before the block was recognized, allowing them to continue. In addition, both students noted how the audio feedback provided by the application was useful in indicating when an action was completed. However, one student suggested providing more feedback to the user in the form of a status bar which he felt would be useful in letting him know whether the application was waiting for input or busy trying to recognize a block.

The students made suggestions that we already considered as features to include in the next iteration. One student wished to be able to modify more than one picture in the same session. We were planning to implement this feature using an RFID reader to sense tagged photos but were unable to do so before this user study and decided to implement it in our next prototype. One student also suggested it would be more convenient to the user if the modified image was projected on the workspace. He suggested this would cause the user to look back and forth between the two displays less often. When in the design phase, we had thought about this feature, but felt it was not part of the basic functionality of our system.

Overall, both students said they would be interested in using future prototypes of our physical chaining

application. Although they agreed that it may be faster to perform similar tasks with Photoshop, they felt it was an interesting application because it was a new way to perform image manipulation.

ANALYSIS

In this paper we have described the design and implementation of function composition in physical chaining applications. This interaction technique raises the ceiling of input for physical chaining applications by assisting the user in dealing with the physical constraints of such applications. By allowing the user to collapse a chain of functions into one physical icon, function composition allows for more optimal use of the physical workspace and facilitates reuse of the application’s physical icons.

We believe function composition is not only useful in an imaging application, but instead is an appropriate interaction technique for a variety of additional applications. In each domain, functions to modify the data source can be defined. Thus a chain of functions could be applied to the data source and saved by the user if it should produce a desirable result. Potential domains include:

Music Editing

Several functions can be defined for editing a music file or score such as transposing the music (changing the key of the musical piece), adjusting the tempo, moving the melody up or down and octave, or modifying the volume of a particular instrument group (e.g., lowering the volume of percussion instruments, raising volume of woodwinds).

Text Document Editing

Within a text document, a user may want to experiment with different fonts, font sizes, spacing settings, etc. In a physical chaining application with a text document as its data source, a function can represent one of these attributes and can then be chained to apply them simultaneously in the document. Pleasing configurations can be saved in one phicon to define a “document style”, similar to the feature supported in Microsoft Word.

Physical Query Language

If the data source is a table from a database, or an entire database itself, each field in one of the tables can be represented by a phicon. A chain of phicons can grow out of each field as the user may want to specify for each field a certain value to search for (e.g., name=“bob”) or set constraints on the search for this field (e.g., NOT name=“bob”, value > 90). With the possibility of non-linear chaining (each “field” phicon has its own chain listing constraints) and the potential use of similar phicons (e.g., NOT, AND, <), such an application would benefit from being able to collapse portions of a query into a single phicon. This may aid in visualizing complex queries.

FUTURE WORK

Future work includes building a second prototype of our application, incorporating the feedback from our informal user study. Once completed, we plan to conduct a formal evaluation of our system. Our current user study design includes allowing test subjects to use the application in two experiments, each using a different set of participants. In the first experiment, subjects will be briefed about the features of our application, giving no special emphasis or encouragement to use the save blocks. They will then be given a list of tasks to perform with our application. Through direct observation, we will record the number of times the subject used function composition and interview the subject afterwards to ask why or why not the subject took advantage of the application's function composition capabilities. In the second experiment, subjects will first be given a list of tasks but will not be able to use function composition. In the second half of the experiment, they will be given the same list of tasks and then introduced to the function composition feature of the application. Observation and interviews would then be used to determine whether having access to function composition positively or negatively affected the user's experience with the application. Possible metrics to measure this include task completion time, number of times function composition was used, and feedback from the subject about using the feature.

Another next step involves integrating function composition into other applications, examples of which are described in the Analysis section. An informal field study can be used to guide us in deciding which application domain or which data source would be best supported with our interaction technique. In this field study, we would investigate the role function composition currently plays in commonly used software applications. Starting from a list of applications which support function composition on a GUI, we can interview users of such applications. These interviews could provide insight into whether or not function composition is used, why it is useful, and what limitations its use imposes on the end-user.

When constructing a chain of functions to act on a single data source in a physical chaining application, the functions are applied to the entire data source. An extension to our imaging application and future applications incorporating function composition would include allowing for selection of a portion of a data source within the chain. A first implementation could use pre-defined selection areas (e.g., top/bottom half, right/left half, quadrants, etc.). A more versatile implementation would allow the user to define the

selection area by directly manipulating a projected image of the data source.

ACKNOWLEDGEMENTS

We would like to thank Scott Klemmer for his assistance with using the SMART Board and his Papier-Mâché toolkit. We would also like to thank our CS260 class for their feedback during our in-class project critique.

REFERENCES

1. Adobe Systems Incorporated. Adobe Photoshop 7. <http://www.adobe.com/products/photoshop/main.html>.
2. Gorbet, M. and Orth, M. (1997). Triangles and The Digital Veil, *Fleshfactor: Informationsmacschin mensch (1997 Ars Electronica Festival Catalog)*, Linz, Austria, SpringerWein-New York, 280-283.
3. Gorbet, M.G., Orth, M., and Ishii, H. Triangles: Tangible Interfaces for manipulation and exploration of digital information topography. In Proceedings of CHI '98 (Los Angeles, CA, April 1998), ACM Press, 49-56.
4. H. Suzuki and H. Kato. AlgoBlock: a tangible programming language, a tool for collaborative learning. In *Proceedings of 4th European Logo Conference*, pages 297-303, August 1993.
5. Ishii, H. and Ullmer, B. (1997). Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms. *Proceedings of Conference on Human Factors in Computing Systems (CHI '97)*, Atlanta, Georgia, ACM Press, 35-42.
6. McNerney, T. (2000). Tangible programming bricks: An approach to making programming accessible to everyone. *MS Thesis, MIT Media Lab*.
7. SMART Technologies, Inc. Rear Projection SMART Board Interactive WhiteBoard. <http://www.smarttech.com/products/rearprojection/index.asp>
8. System Watt Inc. ROBOCUBE. <http://www.watt.co.jp>
9. Terry, M. (2001). Task Blocks: Tangible Interfaces for Creative Exploration." In *CHI 2001 Extended Abstracts*