TRANSFORM As Tangible Programming Environment

Abstract
Currently, computer programming is a skill with a high initial learning curve, which can often discourage early learners. Like any language, however, it is easiest to learn at a young age; thus, numerous attempts have been made to make programming more approachable, interactive, and engaging. We present a vision for a tactile, holistic programming environment, implemented on the TRANSFORM shape display. This environment permits the creation of simple programs using basic function primitives represented in three “levels of abstraction,” all implemented through TRANSFORM’s actuated pins. The result is a programming environment that can be physically manipulated at multiple scales and conceptual levels, leading to a more resilient understanding of computational concepts.

Author Keywords
shape display; tangible programming; tangible UIs; programming languages; visual coding; radical atoms; algorithmic thinking; computational thinking; children; learning; education.

ACM Classification Keywords
H5.2. Information interfaces and presentation (e.g., HCI): User Interfaces
Introduction
In a 2015 interview, President Obama stated: “We’ve got to have our kids in math and science, and it can’t just be a handful of kids. It’s got to be everybody. Everybody’s got to learn how to code early.” [13] Not only have programming skills become essential to working in almost any scientific research field, but the act of learning to program also enhances higher level algorithmic and computational thinking that can be applied to a wide range of problems from both outside and inside the realm of computer programming.

Despite efforts like “Hour of Code” and the recent increased emphasis coming from both grassroots organizations and government agencies, computer science is still seen as a discipline that is inaccessible to many people. Developments such as MIT’s Scratch have dramatically decreased the learning curve for learning programming but still retain the disadvantages of screen interfaces.

Our project aims to seamlessly couple physical interaction with a shape-based programming language to make learning code a tactile experience. It takes advantage of research on tinkering (an exploratory, iterative style of learning [11]), bricolage (combining and recombining a closed set of materials [9,10]), and the benefits of giving users a dynamic “object to think with” [9,10,14]. Tinkering and bricolage are both tactics constantly employed by professional programmers, and our design lets users modify and play with code in real time to encourage these types of interactions. By creating physical shapes to visualize code in three-dimensional space, users gain the added benefit of tangibility, which can help clarify thought and solidify learning.

Related Work
The ongoing efforts by the Tangible Media Group at the MIT Media Lab reflect a strong belief in the potential for enhanced learning and interaction through tactile computing. In addition, the group’s vision of Radical Atoms provides a conceptual framework for the future of material interaction, where material form and appearance are dynamic and reconfigurable. Specifically, the use of shape displays represents a form of human-computer interaction that is grounded by tangible interaction and yet affords a radical layer of computation and digital information. This work builds upon that foundation.

In addition, we review several other implementations of tangible programming languages, dating back to seminal work done by Seymour Papert in 1967:

- Logo (Papert) [9]
- AlgoBlocks (Suzuki, Kato) [12]
- Tangible Programming Bricks (McNerney) [8]
- Tern/Quetzal (Horn, Jacob) [5]
- TurTan (Gallardo, Julia, Jorda) [4]
- E-block (Wang, Zhang, Chen) [15]

Our work differs from these implementations primarily in our use of TRANSFORM. With this, we make use of the dynamic affordance offered by shape displays, thus allowing us to design a programming language that is both tangible and radical simultaneously.
Prototype and Technical Design

Transform
We had access to the TRANSFORM table and programming environment through our participation in Hiroshi Ishii’s [6] Tangible Interface Course at the MIT Media Lab. TRANSFORM is a large table containing three shape displays. Each display contains a grid of plastic “pins” with 16 columns and 24 rows for a total of 1,152 independently addressable pins. Each pin can be actuated in the vertical direction over a range of approximately 3 inches. The entire tangible display is controlled using Objective-C or JavaScript. Additionally, each pin contains a sensor that can detect the current pin location (allowing detection of users pushing down on the pins) and an overhead mounted Kinect display to detect hand gestures in 3D space. [7]

Environment
The tangible programming environment is divided into four "displays": one portable "toolbox" of shape primitives plus the three shape displays of TRANSFORM.

The toolbox, represented as an app or program on a tablet computer carried by the user, acts as a repository for all of the functions that are available in the programming environment. These are displayed as graphic icons with associated names. The user is able to add to this toolbox as he or she defines custom functions and wishes to save them for later use. Selecting items from this toolbox activates them on the first shape display.

The first shape display is for function manipulation. Having chosen a particular function, the user can then perform relevant operations on it (such as rotating its direction, scaling its size or duration, etc.). These are implemented through both simple gestural commands directly above the shape display and physically interacting with the pins. The functions are displayed as graphical icons, similar to the toolbox, using elevated shape display pin heights.

The second shape display is for the assembly of source code. As functions are added from the toolbox (and displayed on the function manipulation display), they are simultaneously represented as individual "lines of code" on the source code display. This allows the code to be assembled in a fashion similar to traditional programming, allowing for spatial efficiency and strengthening cognitive association with GUI-based programming.

The third and final shape display presents the user with the output or result of the program. This content is also rendered immediately, so that any change in output brought about by the addition of a new function from the toolbox is displayed instantly on all three shape displays. A "blinking cursor," represented by a single pin gently oscillating its pin height, informs the user of where on the output each function will act.

Available Functions
The toolbox presents the user with typical functions available in programming languages. In addition, our toolbox contains functions specific to shape displays. For instance, the toolbox contains the functions "Pin Up" and "Pin Down," which command the program to raise or lower sets of pins (indefinitely, until affected by subsequent processes), respectively. In addition, the toolbox contains the function "Move Cursor," which
allows the user to control the exact location on the output where certain functions will be implemented.

Multi-modal Representation:
Three Levels of Abstraction
As discussed, the shape displays present information in three distinct formats: First, diagrammatic unit representations of individual functions; second, assemblies of simplified sections of code; and third, resultant outputs of function actions. As the program runs, the user is able to see which line of code is being implemented (second display), what function that line represents (first display), and what the subsequent output is (third display).

Bi-directional Programming
Each interface acts as both input and output. In addition to visually taking stock of the current step in the program (at all three levels of abstraction), the user can physically interact with the shape display in order to manipulate or delete certain portions of code. Acting on one display results in changes across all three displays. In this way, the user is able to make decisions (and subsequent code changes) about the program at all levels.

Scenario Applications
This environment could provide an excellent format for debugging. As the code encounters certain impossible or nonsensical combinations of functions, the relevant lines of code “blink” through oscillations of the pins. Alternatively, as the user attempts to remove certain functions, the shape displays could “blink” the functions that would be subsequently affected by the deletion.

As a tangible programming system oriented towards teaching computational concepts to children, this environment provides the opportunity for learning from both holistic and detail-oriented perspectives. Some learners may approach the process in a bottom-up fashion, analyzing the individual functions and building up an understanding of their meaning through deliberate manipulation and iteration. Others may prefer a top-down approach, starting with a pre-made program and touching certain parts of the output to see what they are and how they were made. In promoting multi-modal representation and bi-directional programming, we provide alternative learning paths for children of all types.

Conclusion & Future Work
There are many important steps to take in realizing this vision. The basic set of functions that is presented in the toolbox would need to be expanded. A comprehensive base should include standard programming functions (and associated icons) for:

- CONDITIONALS
- ARRAYS and other data structures
- MATH
- RANDOMIZATION

Additionally we should include functions that allow users to manipulate the height of pins in the program result in richer ways. In our prototype we allowed users to control pin height in a coarse manner, either fully up or fully down. A more robust setup would allow users to control pin height in a continuous manner. It could also be useful to provide users with higher-level abstractions for manipulating height such as 2.5D mesh manipulation or extrusion. Future iterations of this
project would more fully explore the advantages of visualizing code in three dimensions.

As longer code is assembled, the middle shape display would need to become "scrollable" in order to allow for additional lines. Alternatively, instead of employing lines to represent individual functions, the display could use individual pins. Perhaps, once the display runs out of available lines, it converts to individual pins (2D to 1D), prompting the user to consider the length of the code and the importance of efficiency and code simplicity.

Expanding the sensing and output capabilities of the TRANSFORM table would allow for additional explorations. Currently tactile user presses can only be sensed in the vertical direction. Adding touch or force sensors on the four vertical faces of each pin could allow for richer tactile interactions including detection of the user's intent to push, slide or rotate objects. Adding overhead projectors to TRANSFORM would allow us to communicate additional details to the user through shape, text and color. With projection, for example, on the center display we could differentiate different code blocks through the use of color or icons. Additionally we could add functions that allow users to manipulate and use colors in their program results.

In this paper we outlined one possible vision for creating a programming language for the TRANSFORM shape display. We built a set of predefined routines to demonstrate future functionality as well as produced a conceptual video to communicate our concept. In future work we plan to build functional prototypes building on our current concepts. Testing with users will allow us to gauge the effectiveness of our approach and generate new tangible programming approaches. In order to facilitate development we wish to streamline technical process required for researchers to approach TRANSFORM and use our tangible programming environment.

Acknowledgements
We thank all our supporting colleagues at the MIT Media Lab, who have provided us the insights and suggestions that enabled us to realize this project. Special thanks goes to Hiroshi Ishii for his leadership in the Tangible Media Group, as well as Daniel Leithinger and Sean Follmer for their initial work in developing shape displays.

References


