

# Task Blocks: Tangible Interfaces for Creative Exploration

Michael Terry

Everyday Computing Lab  
GVU Center, College of Computing  
Georgia Institute of Technology  
Atlanta, GA 30332, USA  
mterry@cc.gatech.edu

## ABSTRACT

This paper introduces Task Blocks, a system that uses physical blocks, called task blocks, to represent computational functions. Users string task blocks together to create a “pipeline” that sequentially manipulates data. Input devices attach directly to individual task blocks to control the effect of each function in the pipeline. The design of the system encourages hands-on, active experimentation by allowing users to directly insert, delete, or modify any function in the pipeline. This paper presents the design of Task Blocks and results from initial prototyping efforts.

## Keywords

tangible interface, creative expression, input interfaces

## INTRODUCTION

Computational technologies offer new possibilities for creative expression, but often stifle fluid interactions by employing standard mediators for users, such as a keyboard, mouse, and WIMP interface. Furthermore, while the creative process emphasizes experimentation and the exploration of possibilities, current computer applications do not take full advantage of computational resources and capabilities to facilitate and enhance these important activities. For example, while undo is a standard feature of today’s software, most applications constrain wider exploration and experimentation by offering only a simple, stack-like undo mechanism that requires users to return to a previous state to “tweak” a function performed in a prior step. Great opportunities exist to create novel interfaces and applications that encourage a more engaging creative experience.

Task Blocks attempts to address these issues with a unique interface that facilitates playing with permutations of data transformations. Task Blocks represents units of computational functionality via tangible objects (task blocks) that are pieced together to create data manipulation pipelines. For example, in an image manipulation

application, a set of blocks could each represent an imaging filter, such as sharpen, contrast adjustment, or “pinch-and-whirl.” Connecting these blocks together creates an image transformation pipeline. Users physically manipulate functions by inserting, removing, or modifying individual task blocks in the pipeline. Input devices attach to task blocks to enable users to directly adjust the parameters to any function in the pipeline with no need to undo to the previous state. Alterations to the pipeline automatically ripple down the pipeline, allowing users to easily test out new possibilities and combinations. The result is a system with a novel interface that encourages active exploration and experimentation by offering users a hands-on approach to transforming data.

## RELATED WORK

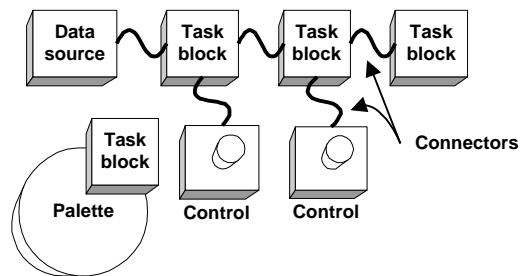
Previous research in tangible media have explored the representation of digital data in physical artifacts. mediaBlocks [1] and Triangles [2] are systems that use tangible objects to represent digital information. These interfaces allow users to both explore data and perform simple data manipulations through compositions of the individual objects. However, the designs of the systems focus primarily on representing data, rather than providing tangible interfaces that afford fine-grained manipulations of data. For example, users cannot directly attach controls to mediaBlocks or Triangles to affect the data represented by the media.

Multiple efforts have created tangible interfaces for program construction. AlgoBlock [3] provides a set of blocks to program a Logo-like language, while a tangible robot programming interface [4] allows users to program Lego Mindstorms™ robots by connecting strings between events and actions. These systems move towards representing computational functionality in tangible objects, but focus on the programming domain.

Finally, analog systems exist that allow users to create the equivalent of data manipulation pipelines. Electric guitar pedals represent one noteworthy example, as they enable musicians to string together adjustable “effects” pedals. An opportunity exists to generalize this model and apply it to the digital realm.

## DESIGN

The design of Task Blocks provides the user with five components: a data source, task blocks, a palette of functions, controls, and a connection mechanism. Figure 1 provides an overview of the entire system.



**Figure 1 - Overview of Task Blocks system**

The Task Blocks system creates a data manipulation pipeline that affects a *data source*. The data source can represent static or dynamic data, such as images or streaming video. To preserve the tangible nature of the system, the data source is represented as a physical block, similar to mediaBlocks [1].

The data source is transformed via *task blocks*. Each task block represents a specific function, displayed in an LCD embedded in the block, which is applicable to the data source. If the data source is a static image, example task blocks may include traditional image filters such as “sharpen,” contrast adjustment, or “pinch-and-whirl.” Functions are dynamically associated with task blocks by the user through a *palette* of functions. A palette is a touch-sensitive LCD tablet displaying specific functions. Users tap task blocks on functions shown on the palette, thus associating the block with the function.

To create the data manipulation pipeline, a *connection mechanism* links the data source with the task blocks, and the task blocks with each other. This mechanism may be realized by designing physically interconnecting blocks, through physical connectors like wires, or other forms that can display, at a glance, the relationship the components have with one another.

*Controls* provide the ability to vary the effect of individual task blocks in the pipeline. Returning to our image example, a control connected to a “sharpen” task block will influence the degree to which the image is sharpened. To accommodate a variety of controls, it is necessary to differentiate between semantically different input devices, for example, a joystick, whose input represents a position in a bounded space, and a mouse, whose input represents a change in position. An input interface standard to task blocks is thus defined. Controls provide their input in one of two forms: as a normalized, relative position between  $-1$  and  $1$ , or as a normalized change in value, from  $-1$  to  $1$ . The first form supports controls with bounded ranges, such

as joysticks, while the second form supports controls that reflect changes in values, such as a mouse. The input interface standard also specifies that either form of input device can be connected to a task block at any time so that the user can choose the input device most appropriate to the situation. A user can therefore adjust the effect of a task block with a slider, then switch to an unbounded knob to make fine-tuned adjustments. The user is free to choose the input device and input semantics most appropriate to the task, facilitating the process of creating an input interface that is well-tailored to the application.

## CURRENT STATUS AND FUTURE WORK

Paper prototypes and a limited functionality prototype implementing a subset of Task Blocks’ design have been created to test out features of the design. The prototypes provide informal feedback from users regarding the general design, and have helped to clarify requirements for a system implementing the full design. In particular, the paper prototypes hint at one of the potential drawbacks of the current design. Using tangible objects requires physical space; as the length of the pipeline increases, so does the need for surface area. Implementing a form of function composition (*i.e.*, representing several functions in one block) may alleviate this problem. Additionally, the current design does not allow the system to operate on select portions of the data source. Task blocks that enable selections within the data source may address this problem.

Work continues towards a system and set of applications that fully implement Task Blocks’ design, with a focus on exploring function composition and on enabling transformations of portions of a data source.

## ACKNOWLEDGMENTS

Thanks to Beth Mynatt, Blair MacIntyre, Lena Mamykina, and the Everyday Computing Lab.

## REFERENCES

1. Ullmer, B., H. Ishii, and D. Glas. mediaBlocks: Physical Containers, Transports, and Controls for Online Media. In *Proceedings of the 25th Annual Conference on Computer Graphics*, 1998, pp. 379-386.
2. Gorbet, M.G., M. Orth, and H. Ishii. Triangles: Tangible Interface for Manipulation and Exploration of Digital Information Topography. In *Conference Proceedings on Human Factors in Computing Systems*, 1998, pp. 49-56.
3. Suzuki, H. and H. Kato. Interaction-Level Support for Collaborative Learning: *AlgoBlock* -- An Open Programming Language. In *Proceedings of CSCL*, 1995.
4. Patten, J., L. Griffith, and H. Ishii. A Tangible Interface for Controlling Robotic Toys. In *CHI 2000 Extended Abstracts*, 2000, pp. 277-278.