Hyperform Specification

Designing with Self-reconfiguring Materials

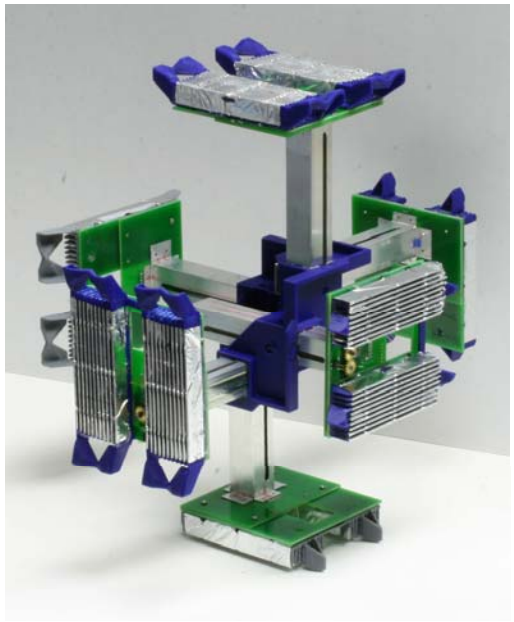

Figure 1: A single prismatic cube hardware module.

Michael Philetus Weller
CoDe Lab
Carnegie Mellon University
Pittsburgh, PA 15213 USA
philetus@cmu.edu

Mark D Gross
CoDe Lab
Carnegie Mellon University
Pittsburgh, PA 15213 USA
mdgross@cmu.edu

Seth Copen Goldstein
Claytronics Group
Carnegie Mellon University
Pittsburgh, PA 15213 USA
seth@cs.cmu.edu

CHI 2009, April 4 – April 9, 2009, Boston, MA, USA
ACM 978-1-60558-247-4/08/04.

Abstract
We describe the current state of our work to develop a *self-reconfiguring material* composed of robotic building blocks, and develop scenarios for the specification of *hyperforms* built with this new type of material that can change in response to tangible and gestural input.

Keywords
Modular robotics, tangible interaction, hyperforms.

ACM Classification Keywords
H.1.2: Models and Principles: User/Machine Systems; **H.5.2**: Information Interfaces and Presentation: User Interfaces: Interaction Styles.

Introduction
Self-reconfiguring materials are composed of ensembles of robotic modules that can reshape their form by downloading and running programs. The prismatic cubes [10] we are building, shown in Figure 1, are an example of such a system. By coordinating with their neighbors the cubes can restack themselves into a nearly infinite number of forms. The cubes are approximately the size of a brick, thus an ensemble of these cubes would be appropriate for applications such as furniture and structures. Another much smaller hardware module still under development, claytronic atoms, or catoms [4], will form a clay-like material.

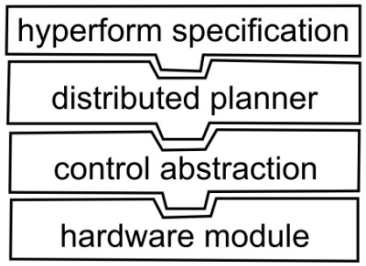

Figure 2: Interface layers for the prismatic cubes.

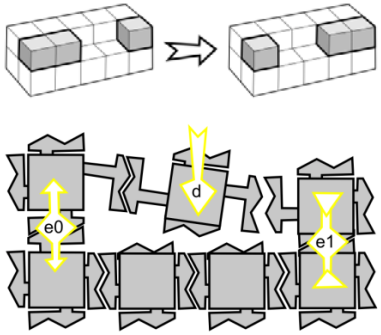

Figure 3: Slide-across rule (top) and control parameters (bottom).

We call the shape of an object composed of self-reconfiguring materials a *hyperform* because it can vary in the four dimensions of space and time. In a world of self-reconfiguring materials the boundary between a design and an object becomes less clear. A single hyperform might express itself as different shapes at different times, and an ensemble's hyperform can be changed just by downloading a new program.

For example, a table hyperform composed of cubes could automatically expand as more people sit down. By making sure there is always one empty seat the table could fold out to make space for additional guests; as people get up from the table after dinner, the table and chairs could fold away leaving only a small dinette with a single empty chair. At the same time, the place settings, if composed of catoms, could coordinate with the contracting table to clear themselves away. The cubes that had been part of the expanded table hyperform could be repurposed to render a sofa and coffee table where guests could relax after dinner.

We are particularly interested in investigating how self-reconfiguring materials can support tangible and gestural interfaces for design. Below we describe our initial work to develop a hardware system that will allow tangible and gestural interaction with hyperforms, as well as our plans for software systems that will enable various modes of hyperform specification.

Hardware and Control Systems

The software architecture for programming an ensemble of cubes has several interface layers. As illustrated in Figure 2, at the top level is a specification of the ensemble's hyperform. A planner takes this specification as input and generates a distributed program to run on an abstraction of the hardware system provided by the control abstraction layer. The planner must be distributed to allow this approach to scale to ensembles composed of many more modules. The control abstraction layer hides the messy details of controlling physical motion. The control abstraction level takes the movements specified by the planner and inserts the low-level instructions required to achieve them on the hardware modules.

To give a simple example, we could download onto an ensemble of cubes the specification for a new hyperform that is just a static shape; e.g., a couch. We would select a from a suite of possible planning methods a distributed planning system such as the Hole Motion Planner [1] or Metamodule Planner [2] to generate instructions to run on the individual modules. As the individual modules each ran their instructions they would reconfigure the ensemble through the application of motion primitives [10], a control abstraction we have developed for the prismatic cubes. With motion primitives each module can choose when to trigger a production rule mapping the current local geometry to a new geometry (Figure 3, top). Triggering a primitive would initiate a series of actuations on the hardware in order to realize this transformation (Figure 3, bottom). Through individual modules firing these motion primitives in a distributed fashion the ensemble would reconfigure into the specified couch shape. Together these layers comprise a complete self-reconfiguring hardware system that can support the hyperform design applications we will discuss in the next section.

Hyperform Design

While there has been a great deal of work on developing hardware modules capable of reliable

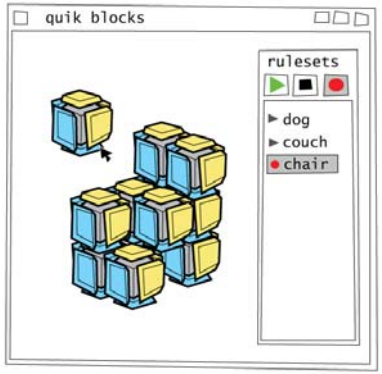

Figure 4: Building a chair in simulation with our ruleset design application.

reconfiguration and distributed planners to allow many modules to collaborate to realize a given shape [11], there has been little investigation of the sorts of tools designers will need to specify hyperforms. Below we describe two scenarios for specifying hyperforms: in the first scenario end-users customize the behavior of a furniture design using gesture and direct manipulation; in the second scenario a combination of a screen-based interface and tangible interaction with the cubes are used to specify goal shapes and constraints on reconfiguration.

Interactive Furniture

Rendering objects in a material with embedded computation, actuation and sensing creates the potential to interact with familiar typologies in new ways. We have chosen a module size that lends itself to furniture. By developing interactive furniture hyperforms we aim to explore several key ideas:

1. hyperforms can respond on their own initiative to changes in the environment;
2. gestures can invoke morphing commands; and
3. end-users can create new commands on the spot by directly manipulating the system.

For example, the expanding table we discussed earlier would be designed to automatically resize itself as new people sit down. However, at times it may be desirable to manually set the size of the table. By making a pulling gesture at the edge of the table it expand in that direction; by making a pushing gesture it would contract. Grabbing a single module in the table and wiggling it would put it into editing mode, allowing modules to be manually manipulated to indicate a new command. For example, the edge of the table could be pulled up to form a lip, and this behavior could be associated with a double knock gesture. Afterward when there was a spill at the table a double knock would raise the lip to contain the spill.

Tangible Specification of Goal Shapes

This scenario showcases the potential for self-reconfiguring materials to be used as an input device as well as a medium for realizing designs. Our ruleset design application, currently under development, will combine a screen interface with tangible input provided by directly manipulating a set of blocks. Goal forms will be demonstrated to the system by stacking the blocks in a particular configuration. The blocks will discover the geometry of their current form and communicate it to the application running on a personal computer. A screen interface (Figure 4) will present controls for compiling a distributed ruleset to produce the current shape, using a modified version of Jones and Mataric's ruleset compiler [5]. Rulesets for different shapes could then be selected and played back. When a ruleset is played, it will be downloaded onto the modules, and each module will initiate movement primitive transitions triggered by the geometry of neighboring modules.

The screen interface will also provide tools for inspecting and manually editing compiled rulesets. The compiler produces rulesets for form-seeking behaviors that converge to a single static form, but the application will also include a variety of rulesets for stable dynamic behaviors, such as those for lateral gaits presented in [3].

Discussion

We have developed the prismatic cubes and motion primitives with the goal of supporting interaction with hyperforms through direct manipulation and gesture. Although we are still developing software systems to

support these modes of interaction, we believe that the interaction models themselves are an important step in thinking about self-reconfiguring materials that informs our work on low-level hardware and control systems.

So far we have produced a handful of prismatic cubes, and due to the technical and financial hurdles involved in mass-producing these modules we expect to build only on the order of tens of these modules on which to run our software in the near future. However we feel that we can make progress by validating these interaction techniques with a combination of experiments on hardware and in simulation. By realizing smaller hyperforms on our limited ensemble of hardware modules we can demonstrate the feasibility of self-reconfiguration and interaction techniques; we can demonstrate the viability of larger and more complex hyperforms in simulation. The development of these new technologies and interaction techniques promises to dramatically increase the responsiveness of the built environment to our ever-changing demands.

Acknowledgements

This research was supported in part by the National Science Foundation under Grants ITR-0326054 and CNS-0428738.

References

[1] De Rosa, M, Goldstein, S C, Lee, P, Campbell, J D and Pillai, P. Scalable Shape Sculpting Via Hole Motion: Motion Planning in Lattice-Constrained Module Robots. *Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, (2006), 1462-1468.

[2] Dewey, D, Srinivasa, S S, Ashley-Rollman, M P, De Rosa, M, Pillai, P, Mowry, T C, Campbell, J D and Goldstein, S C. Generalizing Metamodules to Simplify Planning in Modular Robotic Systems. *Intelligent Robots and Systems (IROS)*, IEEE, (2008).

[3] Fitch, R and Butler, Z. Scalable Locomotion for Large Self-Reconfiguring Robots. *Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, (2007), 2248–2253.

[4] Goldstein, S C, Campbell, J D and Mowry, T C. Programmable Matter. *IEEE Computer*, *38*, 6. (2005), 99-101.

[5] Jones, C and Mataric, M J. From Local to Global Behavior in Intelligent Self-Assembly. *Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, (2003), 721-726.

[6] Karagozler, M E, Campbell, J D, Fedder, G K, Goldstein, S C, Weller, M P and Yoon, B W. Electrostatic Latching for Inter-Module Adhesion, Power Transfer, and Communication in Modular Robots. *Intl. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, (2007), 2779-2786.

[7] Rus, D and Vona, M. Crystalline Robots: Self-Reconfiguration with Compressible Unit Modules. *Autonomous Robots*, *10*, 1. (2001), 107–124.

[8] Suh, J W, Homans, S B and Yim, M. Telecubes: Mechanical Design of a Module for Self-Reconfigurable Robotics. *Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, (2002), 4095–4101.

[9] Vassilvitskii, S, Yim, M and Suh, J. A Complete, Local and Parallel Reconfiguration Algorithm for Cube Style Modular Robots. *Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, (2002), 117–122.

[10] Weller, M P, Karagozler, M E, Kirby, B, Campbell, J D and Goldstein, S C. Movement Primitives for an Orthogonal Prismatic Closed-Lattice-Constrained Self-Reconfiguring Module. *Workshop on Self-Reconfiguring Robots at Intelligent Robots and Systems (IROS)*, (2007).

[11] Yim, M, Wei-Min, S, Salemi, B, Rus, D, Moll, M, Lipson, H, Klavins, E and Chirikjian, G S. Modular Self-Reconfigurable Robot Systems. *Robotics and Automation*, *14*, 1. (2007), 43-52.