

A Flexible Construction Kit for Interfacing with 3D Geometry

Ken Camarata, Ellen Yi-Luen Do, Markus Eng, Mark D. Gross, Michael Philetus Weller

Design Machine Group
University of Washington
Seattle, WA, USA

(kcamarat,ellendo,markuse,mdgross,philetus)@u.washington.edu

ABSTRACT

We describe a framework for computationally embedded physical modeling kits to support interfacing with 3D geometry for science, engineering, and design applications and introduce an example project called FlexM that supports dynamic geometry construction and feedback with a hub and strut physical model.

Keywords

Computationally Enhanced Construction Kits, Tangible UIs, 3D Interaction, Input and Interaction Technologies.

INTRODUCTION

Making and manipulating 3D models with WIMPy graphical user interfaces has a steep learning curve, making these activities prohibitive for anyone other than highly trained designers. We would like to enable ordinary people to work with 3D models without extensive technical training. Fortunately, there is another 3D modeling paradigm that many more people are familiar with. As children many of us played with construction kits such as wood blocks, Tinker Toys, Lego or Meccano. Although most of these construction kits do not provide the level of control and detail that a 3D modeling application would, they allow almost anyone to create a 3D sketch of a physical form. With imagination and perhaps a little extra description these 3D construction kit sketches can more than adequately describe 3D forms such as a building, a molecule or a dinosaur.

We believe that low cost microcontrollers, sensors, and wireless communication now enables a new generation of construction kits, similar in spirit to the popular construction toys of the early and mid-twentieth century, but adding the “magic” of computation. We want to exploit the complementary benefits of creating and working with physical 3-D models and computational enhancements to create more powerful and compelling environments for learning and design.

**LEAVE BLANK THE LAST 2.5cm
OF THE LEFT COLUMN
ON THE FIRST PAGE
FOR US TO PUT IN
THE COPYRIGHT NOTICE!**

Framework

We propose a framework for capturing the configuration and dynamic geometry of construction kits to enable interaction with software applications, building on Eisenberg et al’s notion of Computationally Enhanced Construction Kits [3]. A computationally enhanced construction kit is a conventional kit with microprocessors and sensors embedded in many or all of its pieces so that a physical model can be sensed and reconstructed as a 3D digital model. We distinguish between *configuration* and *dynamic geometry*. By configuration we mean which pieces of the construction kit are connected to which other pieces, and when there are multiple ways to connect two pieces how they connect (figure 1). By dynamic geometry we include also the current state of any moving parts (figure 2), such as a hinge [11]. We see sensing configuration as roughly analogous to compiling a program and sensing dynamic geometry as analogous to supporting its run-time behavior. A computationally enhanced construction kit must have sensors to capture both the configuration and dynamic geometry of models that users make.

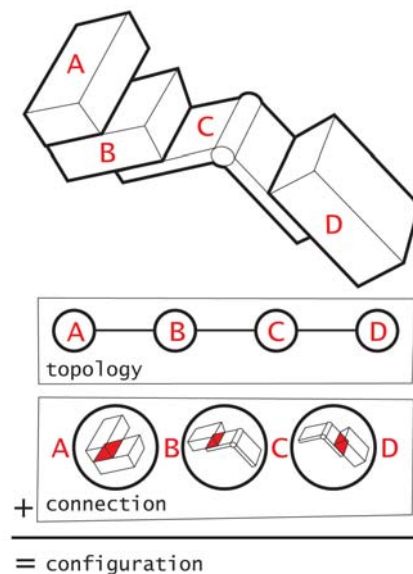


Figure 1: configuration describes which pieces are connected, and how they connect.

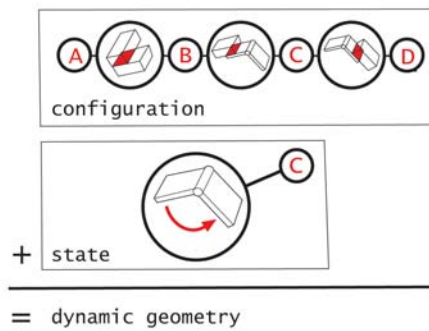


Figure 2: dynamic geometry describes current position of moving parts.

The distinction of configuration and dynamic geometry reflects the familiar construction kit interaction paradigm. The initial stage of interaction involves building a structure and assigning meaning to it. For example a child might declare “I am building a robot out of tinker toys” or “This is a model of hydrochloric acid.”

The following stage involves interacting with the object according to the status that has been attributed to it, “Now the robot is climbing a mountain” or “Now the hydrochloric acid is reacting with sodium hydroxide.” Assigning meaning is important in the first stage because the same abstract physical model can be used to represent objects in widely different domains (as our robot/chemistry examples remind us)—and determines what behaviors the model will support in the second stage, for example, in a related desktop simulation.

The goal of this framework is to allow computationally enhanced construction kits to provide an interface to create rough 3D digital models, attribute meaning, behaviors or additional levels of description to the model, and then to continue using the physical model to interact with its digital representation. In addition to accounting for physical construction kits as input, our framework also encompasses construction kits as output devices. For example lights and speakers built into components allow an application to give feedback through the construction kit; moving parts in the kit can employ actuators to allow a software application to adjust the structure’s dynamic geometry.

As part of an effort to explore this design space of Computationally Enhanced Construction Kits, we have built a working prototype of a computationally enhanced hub-and-strut geometry construction kit, FlexM. Each hub has several sockets to receive struts. The struts are passive but a microprocessor and sensors on each hub detect which other hubs it is connected to and through which socket. Each socket can also be rotated relative to the base of its hub, and sensors allow the hub to measure the current angle of each socket.

Other projects, including those mentioned below in the section on Related Work, have detected the configuration of parts to reconstruct a 3D digital model, or developed specialized kits that capture dynamic geometry to animate

digital models of characters. We believe that there is a need for a more general framework that captures the full representative power of traditional construction kits and computational modeling.

RELATED WORK

Fischer Technik was among the first to enhance a commercial mechanical construction kit toy with computational abilities. Among the best known today is Lego Mindstorms. It provides a microcontroller that end users can program to control motors, lights, and sensors. However, a Lego Mindstorms kit provides only one microcontroller. This predisposes the kit toward a class of constructions in which a single central “brain” controls a model, for example, robot vehicles. Although the separation of computational components (microprocessor, sensors, actuators) permits end-users to combine these elements with physical components, we are more interested in the close coupling or integration of computational and physical components. Construction kits that are computationally enhanced in this way include components that are at once physical/mechanical building blocks and computational ones.

Aish’s Building Block System [1] was a three dimensional block system for inputting architectural models to a CAD system. Frazer’s [5] 3D input devices enabled designers to build models that interface with software that can give design advice. Anderson et al.’s Computational Building Blocks [2] facilitates computer modeling with instrumented snap-together plastic blocks. In Gorbet and Orth’s [6] Triangles, a construction kit of flat plastic triangles that interface to a computer, each triangle tile corresponds to a different application, such as an email client or a personal calendar, or in a later version, a character or object in a story. Mechanical and electronic magnetic connectors allow the user to build a variety of geometric forms that correspond to his suite of applications. Although the Triangles have hinges they assemble to make a static and rigid form. Each of these projects, however, lacks a real-time interface for detecting moving pieces — what our framework terms *dynamic geometry*.

Several projects track movements of physical objects to generate or control animated graphics. Monkey™ is a specialized input device for virtual body animation [4]. It resembles a mechanical mannequin with articulated limbs. Instead of constructing a simulation of human animation and locomotion using a screen interface, the animator poses and moves the Monkey™ to define the character’s animation. Topobo [10], is a construction kit of articulating vertebra-like pieces for building posable forms with embedded kinetic memory. The embedded memory records angular movement at the joints. Users build a creature, move the model across a terrain, and then watch the model replay its movement from its embedded kinetic memory.

Both Phidgets and CUBIK are concerned with controlling computational behavior with physical manipulation. Phidgets, a construction kit of physical computing widgets:

sensors, motors, radio frequency ID readers, and a software interface for user interaction [7] enables end users to assemble hybrid computational-physical devices without knowledge of processors, communication protocols or programming.

CUBIK is a tangible modeling interface to aid architects and designers in 3D modeling. It takes the form of a mechanical cube [8]. The designer manipulates dials on the cube’s face to expand or contract the dimension of a corresponding computer graphics representation. The communication between the GUI and CUBIK is bi-directional: the designer can also manipulate the physical cube through the GUI.

SPECIFICATION

We set out to build a hub-and-strut geometry construction kit, as a prototype system that would require capturing both configuration and dynamic geometry. As its name implies, this kind of construction kit comprises hubs and struts, forming in effect the vertices and edges of a graph. The specific design of such kits varies tremendously, giving rise to a wide range of variants with different properties. For example, TinkerToy’s hubs (wooden spools with radially drilled holes) have fixed connection angles and fixed length rigid struts. In ZomeTools, hubs also determine the angles, but unlike TinkerToy the hub angles are three-dimensional and struts of various lengths are keyed to specific sockets in the hub. In some kits the hubs are made of flexible plastic and rigid struts allowing the model to flex and deform. In others the hubs are rigid but the struts (e.g., made of plastic straws) are somewhat flexible. In a “ball and spring” molecular modeling kit, springs are inserted into holes drilled in color coded wooden spheres at the appropriate bond angles for different kinds of atoms.

We chose to build a hub-and-strut kit with fixed struts and flexible hubs. In the beginning we just wanted to build a physical analog for a 3-D model that we could flex dynamically (hence the name of our prototype, “FlexM”). However, this choice enables us to explore both the fixed and dynamic components of our framework. Also, the graph model of a hub-and-strut kit makes it easy to map the designs to a wide variety of domains.

We set as our goal the design of a kit able to to serve as an input device that can:

- 1) determine the model’s configuration—which hubs connect through which sockets.
- 2) determine the model’s dynamic geometry—how it is flexed.
- 3) send model configuration and dynamic geometry to a host computer for further processing.

In addition, we also want the kit to serve as an output device that can at least:

- 4) highlight parts of the constructed model, perhaps even

- 5) modify (flex) the angles between vertices of the model.

IMPLEMENTATION

We constructed a series of prototypes to explore how to capture configuration and dynamic geometry. Table 1 lists the prototypes, the issues each explored, and the technologies employed.

Table 1: prototypes, issues, and technologies

0	dynamic geometry	surgical tubing, bend sensor, wooden sticks
1	configuration	wooden cubes with lights and photosensors
2	dynamic geometry	bend sensor embedded in silicone mold
3	configuration and dynamic geometry	popsicle stick hinge with sliding potentiometer
4	configuration and dynamic geometry	popsicle stick hinge with rotational potentiometer
5	configuration and dynamic geometry, manufacturability	rapid prototyped plastic hinge with rotational potentiometers

The first prototype (0) we used to demonstrate the concept was a cube made of thin wooden (shish kabob) sticks and surgical tubing, with bend sensors inserted to sense when the cube was deformed. We used a microcontroller (first an MIT Cricket, subsequently a Handyboard) to measure variable resistance of the bend sensors, and drive the display (in VRML) of a three-dimensional model of the cube. This prototype only sensed geometry, and it was not modular: one could not disassemble and reconfigure the components, in part because it was difficult to work with the sticks and tubing without disturbing the bend sensor. Also, the bend sensor is relatively expensive, tends to perform differently over time (with fatigue), and each unit performs differently, requiring careful calibration.

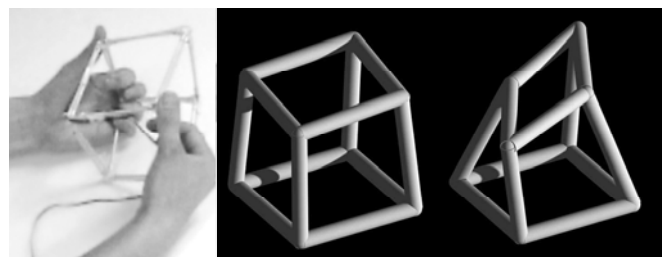


Figure 3: First surgical tube model deforming a computer-graphics cube.

Our current working prototype uses a combination of high-intensity LEDs and photosensors to determine model

topology, rotational potentiometers to determine model geometry; and a microprocessor with a radio transceiver to send information collected at each hub to a central base station that assembles the information received and passes it along to a desktop computer.

Mechanics

We explored several variations of the mechanical design of the hubs, following the initial stick and surgical tubing prototype. We tried casting bend sensors into a silicone hub, which made a flexible cast-in-place hub. We built a rigid hub design that accepts struts into sockets in the faces of a cube; we used this prototype to develop the topology-sensing technique, but the rigid connections violated our specification for flexible hubs. For our first working prototype that sensed both configuration and dynamic geometry we settled on a mechanical hinge design somewhat like an umbrella. Each socket is mounted at the end of two popsicle-stick shaped pieces of wood (1 cm x 10 cm) that are hinged along their long edges. Our prototypes 3 & 4 (figure 4, 6, and 7) have three of these hinged pairs, which allows the hub to flex from flat (120° between edges) to closed (almost 0° between edges).

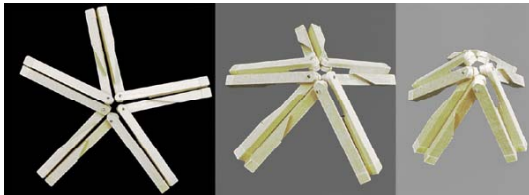


Figure 4: “Popsicle-stick” mechanical hinge design.

Configuration

To determine the model configuration, the base station signals each hub one by one to turn on its LEDs. The bright light at the end of each of the sockets shines along the length of the acrylic rod, and photocells in the sockets of any connected hubs can sense it (figure 5). The base station polls all the other (unlighted) hubs to determine which of them are connected to the currently lighted hub and through which socket. When the base station has finished lighting and polling hubs, it has built a table of connections that taken together represent the model’s topology. The following pseudocode illustrates the algorithm.

```

for lighted-hub in hubs
  {tell lighted-hub “light on”
  for each hub in hubs
    {for each socket in hub.sockets
      {ask socket.lightsensor “sees-light?”
      append (hub socket lightedhub) to connections}}
  tell lighted-hub “light off”}
return connections

```

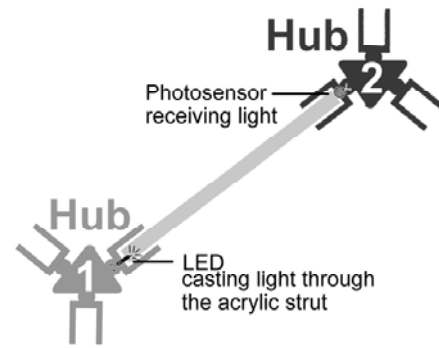


Figure 5: Optically sensing configuration: LEDs in sockets of Hub #1 are sensed by photocells in other hub’s sockets.



Figure 6: Clear acrylic struts connect three hubs.

We considered using electrical connections (e.g., using an audio mini-jack connector) rather than optical ones, although we were concerned about retaining a positive connection as users flex the model. The sequence of flashing LEDs is visually attractive and also reveals the configuration-sensing algorithm. Moreover, we can use the LEDs in each socket as an output medium, to highlight selected portions of the model.

Dynamic Geometry

After abandoning bend sensors as costly and difficult to calibrate we used inexpensive potentiometers to measure the flex angle of the hubs. In the first model we mounted a sliding potentiometer to measure the separation between each pair of arms (figure 7 and 8 – left). This works well to measure the planar angle but the size and form of the sliding potentiometer makes the hub appear unwieldy. In the next version of the hub we switched to rotational potentiometers, with the rotating axis making the hinge between each pair of arms (figures 7 and 8 – right).



Figure 7: Popsicle-stick hub with sliding (left) and rotational (right) potentiometers to measure angle.



Figure 8: Left: sliding potentiometer measures angle; right: rotational potentiometer makes and measures hinge.

Strut length

The configuration and dynamic geometry of a model is determined by the vertex angles between edges, but also by the lengths of struts. Our present system uses only one strut length. However, we have considered several methods for determining strut length, which we plan to explore in the next iteration prototype. The least sophisticated is to physically key the ends of struts and mount pushbutton micro-switches in the sockets, so that each different strut closes a different combination of switches. A variant is to coat the edges of the strut with a pattern of conductive metal or paint, and mount contacts on the inside of the socket. Another method, attractive because it requires no additional instrumentation, would measure the attenuation of light along the length of the strut—the longer the strut, the lower the intensity of light reaching the photo-sensor on the other end. Initial testing suggests that this method is plausible, though sensitive to ambient lighting conditions. A third method uses different tints of plastic for different strut lengths (e.g., red = long; blue = medium; green = short). Then using a color-sensitive photosensor we can determine the color (and therefore the length) of the strut that connects each pair of hubs.

Communication

Our first prototype used an MIT Cricket (a PIC 16F84 microcontroller board with two i/o ports and infrared communication) to read the resistance values of bend sensors. When we moved to potentiometers and added the

photocell/LED combination to sense configuration we needed additional i/o ports, so we began using Fred Martin's Handyboard instead [9]. We wired each hub to an i/o port on the Handyboard, and ran a program on the Handyboard to light and poll the hubs as described earlier; then we sent the configuration and dynamic geometry data along a serial line to a desktop computer for further processing. This configuration allowed us to develop the mechanical and electronic design of the hubs and test the sensing algorithms. The principal drawback of this approach is that the hubs must all be wired to the Handyboard.

To address this drawback we replaced the wired Handyboard with a wireless variation of the communications design. Each hub now contains a Basic Stamp 2 microcontroller (popular among hobbyists) with a Surelink radio frequency transceiver. The transceivers communicate with a single base station unit that, like the Handyboard, polls the hubs and collects the configuration and dynamic geometry data and sends it along a wired serial connection to a desktop computer.

The model on the desktop

After transmitting the model's configuration and dynamic geometry to the desktop computer, the base station continues to poll the construction kit model for changes. As the user flexes the model, altering the potentiometer values that measure angles at the vertices, and occasionally reconfiguring the strut connections among hubs, the base station sends updates to the desktop computer. A receiving application on the desktop computer constructs a software model of the physical construction that describes the hubs, their connections among sockets, and angles.

The software model is simple. Each hub maintains a table of its sockets which in turn maintains its socket angle. Each socket also maintains a "connected-to" slot, which may point to a strut object, if there is a strut connecting that socket to a socket in another hub. Each strut simply points to both the sockets that it connects. This graph data structure of hubs, sockets, and struts completely describes the configuration and dynamic geometry of the model. To draw the model in 3-D on the screen, for example, a display routine traverses the graph breadth first using hub angles to determine the position of each next hub as it proceeds.

Diagnostic software shown in Figure 9 (left) reveals how flexing a hub controls a computer graphics model and (right) the socket-to-socket connections among four hubs. The abstract data structure that describes the model's configuration and dynamic geometry is then made available to domain-specific software applications, such as those identified in the following section.

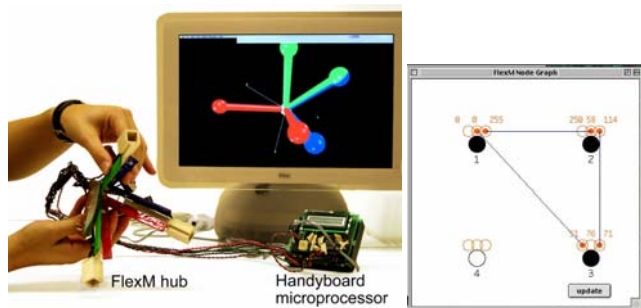


Figure 9: driving computer graphics models with configuration and dynamic geometry from a FlexM construction.

Beyond merely reflecting the 3-D model structure on the screen, the graph structure built on the desktop computer is sufficient to interface to a variety of applications. For example, by identifying different hubs as different atom types, (which we can do when we load the microcontroller code), we can use the graph to provide input to a ready-to-hand molecule visualization program such as SMILE or CHIME (<http://www.umass.edu/microbio/chime/>).

APPLICATIONS

A construction kit has various applications beyond a toy. Architects use physical models to think about the spatial characteristics of a building. Mechanical and civil engineers make physical models of kinematic linkages and structural support systems. Chemists and biologists build physical models to think about the three-dimensional structure of molecules and proteins. Physical models complement computer graphics models and performance simulations by providing users (designers, scientists, engineers) a kinesthetic (literally a “hands on”) sense of the three-dimensional structure of the artifact in question.

An essential part of our framework is the idea that, despite the concreteness of physical construction kits, they serve as abstract representations to interact with domain-specific applications. Thus, a hub-and-strut kit can serve to model molecular models, structural members of a truss or space frame, or a kinematic model of a legged animal or robot. It is essential—though perhaps counterintuitive—not to fix the mapping from the abstract (though concrete) physical model to the specific application domain. However, we mention several candidate applications for the FlexM hub-and-strut construction kit.

For example, an engineering student builds a model of a truss or space frame. Then, because the elements of the model are computationally enabled, the truss geometry is captured and transferred to a structural analysis simulation, running on a desktop computer nearby. The simulation results are displayed on top of a computer graphics model of the physical truss. As the student loads and twists the truss in different ways the simulation responds immediately. The truss itself indicates compression and tension forces by lighting up its members in different colors.

A chemistry student builds a model of a sugar molecule. As the student constructs the molecule, the model could highlight active sites for bonding. As with the structural model, a nearby desktop computer offers additional information about sugar and related molecules; for example giving information about stereoisomers, and other geometric variants.

Sharing models over the Web is another interesting extension. Students can upload models they have built onto a shared Web site, much as gamers upload characters and scenes they have created. Because the models are computationally sensed, they (or at least their virtual representation) can easily be transferred without the student having to photograph or scan them.

FUTURE WORK

Rapid Prototyped Hubs

Our first several rounds of prototypes were hand-crafted, as we explored different variations of sensing technologies. However, we have designed a hub we can print directly to ABS plastic with a form deposition printer. Figure 10 shows a three-socket version of this hub. This hub design differs from the earlier handmade hinged arm prototype. In the new model potentiometers measure the angle of each edge of the model (strut) from the plane of the hub; whereas in the earlier hand-made hub, the potentiometers measure the angles between edges (struts).

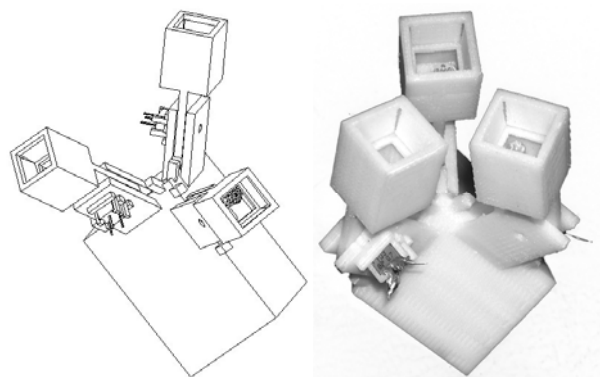


Figure 10: direct-to-plastic rapid prototyped hub: CAD model (left); physical component (right). The large rectangular form at bottom mounts the microcontroller, power supply, and radio frequency transceiver.

Smaller Footprint

We originally chose the Basic Stamp 2 for the hub microcontrollers because it offers a simple programming platform suitable for novice programmers, and we want to leave open the option for end users to program the hubs. However, our current microcontroller-RF transceiver combination is large, relative to the size of the rest of the hub. A future version of the hubs will employ a more compact microcontroller and RF transceiver, such as the Chipcon 1010 that combines these two functionalities in one IC. This will enable us to reduce the size of the computational component and its associated power supply from the current 9 volt battery to a coin-size cell.

More sensors and actuators

We plan to add orientation sensing to the hubs so that the model can transmit information about the overall orientation and the relative orientation of model parts. Also mentioned earlier, we would like to sense the lengths of struts that connect the hubs. In addition, we plan to add small motors to the hubs so that the software application can drive the dynamic geometry of the model; a challenge that we are currently addressing in a related project on modular robotic building blocks.

DISCUSSION

As microcontrollers, sensors, and wireless communication continue to become cheaper and smaller we expect to see a new space of computationally enhanced construction toys, that comprise not only a single microprocessor per kit, but in which each component may employ sensors, actuators, and microprocessors, and communications.

Our hub-and-strut kit, FlexM, is one point in the design space of such kits. We have described a framework in which to locate kits like FlexM. We are more generally interested in exploring different kinds of construction kits, and the ways in which they can sense and actuate configuration and dynamic geometry. As we continue to refine the design of FlexM, we also hope to apply the lessons learned to other computationally enhanced construction kits. For example, we plan to apply the sensing schemes we used in FlexM to build other physical modeling kits for applications such as modeling architectural and mechanical structures.

The two stage interaction described by our framework establishes a clear paradigm for the functionality to be supported by future computationally enhanced construction kits. Components with improved support for form generation, the attribution of meaning and behaviors, and interaction with digital representations will better harness the power of physical construction kits to create a rich and responsive experience. Beyond directly applying the sensing techniques employed in FlexM, we are also excited to explore the possibilities of using models as output as well as input media—touched on in the lighted hubs and struts in FlexM.

Even more speculatively, we are intrigued by the idea of treating hybrid physical/computational components as elements in a language of mechanical and computational capabilities. What would be the process of specifying a design? How might we build compilers and generators for these hybrid components? Finally, how might we make this process accessible—and useful—to end-users? We hope to take up these challenges in a future stage of this work.

ACKNOWLEDGEMENTS

We thank toy designer and inventor Don C. Witte, who helped with early discussions and suggested the LED/photosensor method of capturing model configuration, and James McMurray of the UW Metals program in the School of Art, for help with rapid prototyping.

This research was supported in part by the National Science Foundation under Grants CCLI-0127579 and ITR-0326054. The views and findings contained in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

1. Aish, R. 3D Input for CAAD Systems. *Computer-Aided Design*, 11, 2 (1979). 66-70.
2. Anderson, D., Frankel, J., Marks, J., Agarwala, A., Beardsley, P., Hodgins, J., Leigh, D., Ryall, K., Sullivan, E., Yedidia, J. Tangible Interaction + Graphical Interpretation: A New Approach to 3D Modeling. in *SIGGRAPH 2000*, ACM, 2000, 393-402.
3. Blauvelt, G., Wensch, T. and Eisenberg, M. Integrating Craft Materials and Computation. *Knowledge-Based Systems*, 13, 7-8 (2000). 471-478.
4. Esposito, C., Paley, W. B., and Ong, J. Of mice and monkeys: A specialized input device for virtual body animation. in *Symposium on Interactive 3D Graphics*, Monterrey, 1995, 109-114.
5. Frazer, J., Frazer, J., and Frazer, P. New developments in intelligent modelling. in *Computer Graphics 81*, Online Publications, 1981, 139-154.
6. Gorbet, M. and Orth, M. Triangles: Design of a Physical/Digital Construction Kit. in *Designing Interactive Systems (DIS-97)*, ACM, 1997, 125-128.
7. Greenberg, S. and Fitchett, C. Phidegts: Easy development of physical interfaces through physical widgets. in *UIST 2001 Symposium on User Interface Software and Technology*, ACM, 2001, 209-218.
8. Lertsithichai, S. and Seegmiller, M. CUBIK: A bi-directional tangible modeling interface. in *Human Factors in Computing Systems, CHI 2002*, 2002, 756-757.
9. Martin, F. *Robotic Explorations: A Hands-On Introduction to Engineering*. Prentice-Hall, Upper Saddle River, New Jersey, 2001.
10. Raffle, H., Parkes, A. and Ishii, H. Topobo: A constructive assembly system with kinetic memory. in *Human Factors in Computing (CHI) '04*, ACM, 2004.
11. Wensch, T. and Eisenberg, M. The programmable hinge: toward computationally enhanced crafts. in *UIST 1998*, ACM, 1998, 89-96.