The Programmable Hinge: Toward Computationally Enhanced Crafts

Thomas Wrensch and Michael Eisenberg
Department of Computer Science
Campus Box 430
University of Colorado, Boulder CO USA 80309-0430
wrensch@cs.colorado.edu; duck@cs.colorado.edu

ABSTRACT
Traditionally, the practitioners of home crafting and the practitioners of computing tend to occupy distinct, non-overlapping cultures. Those small, ubiquitous items of the crafting culture—string, thumbtacks, screws, nails, and so forth—thus tend to be viewed as inevitably "low-tech" objects. This paper describes our initial efforts toward integrating computational and crafting media by creating an instance of a *computationally-enhanced craft item*: a programmable hinge. We describe several prototype models of the hinge; outline a sample project in which the hinge might be employed; and discuss a variety of fundamental issues that affect the design of computationally-enhanced craft items generally.

Keywords
Computationally-enhanced crafts, programmable hinge, integration of physical and computational media, crafts.

INTRODUCTION

There is an outdoor mall not too far from our campus—a fairly typical specimen. Besides the usual supermarket, drugstore, and video rental store, there are two rather interesting, and well-stocked, retail outlets: a crafts store and, at not too far a distance, an electronics store. The former features the type of materials that one reads about in crafters' magazines: fabric, adhesives, paper in a huge variety of grades, sewing materials, paints, scissors, dowels, beads, and so forth. The latter focuses on electronic equipment of various kinds, prominently including home computers and peripherals.

Both stores appear to be doing a thriving business, but despite their geographic proximity they represent distinctly different worlds. The respective cultures of crafting and computing—the populations of customers in the two stores, the types of projects that those customers undertake, the magazines they read—seem to have almost no overlap.

UIST '98. San Francisco, CA

On the one hand, this separation of cultures might not be too surprising: people's interests do, after all, vary. And yet it seems to us that there are interesting opportunities for integration that are missed in this cultural divide. The world of crafting—while it has experienced a startling expansion in available materials and techniques over the past several decades—could be vastly enriched by the inclusion of appropriate computational devices, languages, and environments. The world of computing could benefit, we believe, by embedding computation in the huge array of low-cost material substrates available to the home crafter. A systematic detente—an effort to rethink the objects of crafting by seeing when and where they may be endowed with computational capabilities—can only expand the range of expression of both cultures.

This paper is a discussion of one, still rather early, effort in this direction: a "programmable hinge". We see this as a single illustrative instance of a much larger class of potentially viable computationally-enhanced craft items. Just as hinges can be made programmable, so too (we feel) can many of the other small, ubiquitous items of crafting—string, thumbtacks, screws, perhaps even adhesives and coatings. Moreover, "programmability" for such materials and objects takes on a particular meaning and style suited to the larger purpose of crafting. That is, by making a craft object such as a hinge "programmable", we mean endowing it with some (usually a very modest) amount of computational power—perhaps the ability to read a few bits' worth of sensor input and/or execute a short program. Such objects should retain, wherever possible, the flexibility and low cost characteristic of their nonprogrammable analogues.

The remainder of this paper will focus primarily on the programmable hinge, using it as a springboard for discussing some of the more general issues in the design and use of computationally enhanced craft items. The following (second) section presents several successive working prototype designs for the programmable hinge, outlining the strengths and weaknesses of each. In the third section we outline a sample craft project that could plausibly make use of the hinge. The fourth section discusses some of the recurring (and we believe fundamental) issues involved in creating workable hardware and software for a wide range of craft items,

using the hinge as a source of representative observations. Finally, in the fifth section, we discuss some related work in the integration of computation and materials, and close with some thoughts on the ways in which a proliferation of novel craft items could enrich the reigning culture of computing.

THE PROGRAMMABLE HINGE AND ITS DESIGN

In this section, we present several successive prototype designs of a programmable hinge, contrasting the various examples to highlight crucial issues. Despite their differences, in each case the basic design philosophy is the same: namely to create a hinge whose opening and closing behavior can be dictated by a program associated with and effectively built into the object itself. Currently, all our prototypes make use of the "cricket"—the programmable Lego brick designed by Resnick and his colleagues at the MIT Media Lab[10,11]—as a small standalone computer attached to the hinge; we will return to this choice (and its associated challenges and limitations) very shortly.

Prototype 1: a Mechanical (Motor and Gear) Actuator

The first programmable hinge design is conceptually straightforward but (as we will see) problematic. Figure 1 shows a functional diagram of the hinge's design, and Figure 2 a photograph of the working prototype itself. As shown in Figure 1, the hinge employs a traditional motor-gear system for control. The control block at the left of Figure 1 represents a combination of the cricket, external AA battery, and relays used to drive the motor. The only moving portion of the hinge is the flap attached to the shaft at the upper right of Figure 1: while the control block, motor and gearbox are stationary, the flap rotates out of the plane of the figure (this is thus a single-flap hinge as opposed to a design in which two flaps open and close). The program embedded within the control block's cricket may be used (e.g.) to rotate the hinge flap at fixed intervals, or in response to input from an additional sensor.

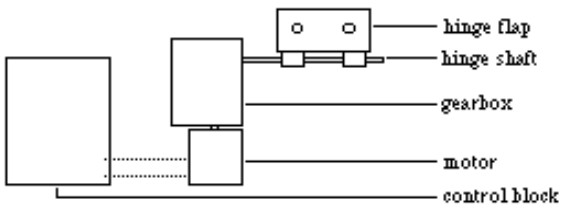

Figure 1. Diagram of the first prototype hinge.

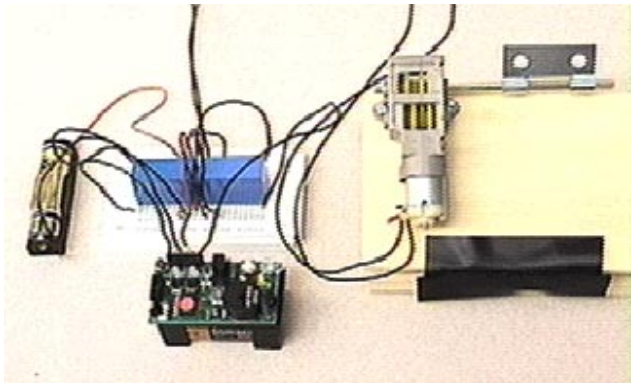

Figure 2. A photograph of the working prototype. The "control block" of Figure 1 is visible toward the left of the photo (the cricket is seen toward the bottom left, the AA battery is at extreme left, and the relays are visible just above the cricket). The hinge flap and shaft are at the upper right of the photo.

This hinge design is reasonable in some respects: the motor and gearbox arrangement may be used to provide torque sufficient for a wide range of crafting projects. On the other hand, the size of the object makes it unwieldy for many applications; the mechanism is obtrusive, producing a somewhat jerky open-and-close behavior; and the general "unhinge-like" look of the prototype renders it aesthetically unsatisfying.

Prototype 2: a Shape Memory Wire Actuator

In contrast to the first design, the second prototype hinge employs shape memory wire as an actuator.[7] This is material that can be "programmed" to adopt a particular shape when it is heated to 50^{o} C. A short piece of wire, a few inches long, can easily be heated to that temperature using resistive heating from a standard AA or 9V battery. Thus, a piece of shape memory wire may be customized so that it will take on a predetermined shape (such as "bent" or "straightened") when heated; when cool, it no longer "remembers" its heated shape and may be bent in much the same fashion as a normal piece of wire.

Figures 3 and 4 comprise a functional diagram of the second prototype hinge, and Figure 5 is a photograph of the working prototype. This new design is a two-flap hinge in which the flaps open by moving apart. The design makes use of the idea suggested in the previous paragraph: two wires are mounted across the hinge flaps as seen toward the right of Figure 3. One of these wires is customized to open when heated (i.e., when a current is run across it); the other is customized to bend when heated. Since the force exerted by a heated wire returning to its programmed shape is larger than the force required to bend a cool wire, an open-close actuator may be built with two lengths of wire—one whose job it is bend and

close the hinge when signalled, and the other whose job it is to straighten and open the hinge. As in the earlier prototype, the control block seen toward the left of Figure 3 consists of a cricket, external battery, and relays used to control the power to the memory wire.

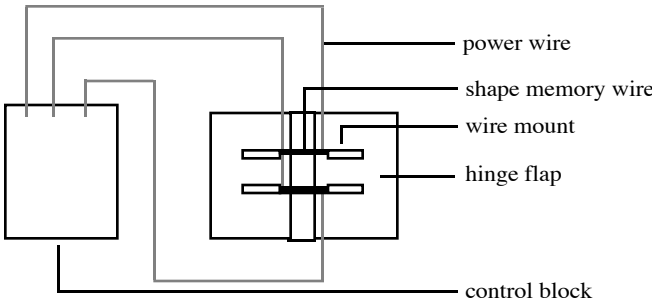

Figure 3. Diagram of the second prototype hinge.

Figure 4. A sketch of the hinge as seen from the top.

At this juncture, it is worth pausing to compare the two prototype designs. Unlike the first design, this second prototype does look and work like a normal hinge; and it has a much smoother (though slower) motion from its open to closed state and back. From a control standpoint the two designs are almost identical, requiring about the same power to operate, although the shape memory wire is much less sensitive to differing voltage levels. The major disadvantage of the second design when compared to the first is in the relatively low torque exerted by the memory wire actuator, and in its slow speed of operation. Indeed, there is an interesting tradeoff between power requirements of the shape memory wire actuator and its cycle time: by thermally insulating the wire, it can be heated to its transition temperature using lower power, but only at the cost of a longer "cool down" time before the opposing wire can be heated and the opposite behavior of the hinge invoked. (Thus, to put the matter pragmatically, the lower-power hinge can only be programmed to open and close at relatively longer intervals.)

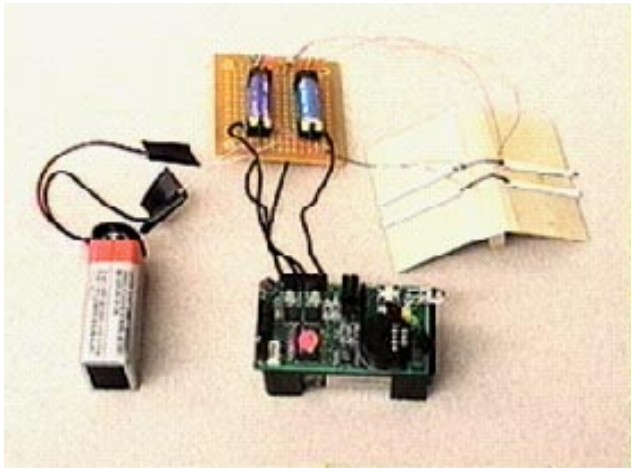

Figure 5. A photograph of the second prototype hinge. Again, the control block is visible toward the left of the photo (the cricket is at bottom center), and the hinge with shape memory wire actuator is seen toward the right

Prototype 3: Twisted Shape Memory Wire

The third prototype hinge again makes use of shape memory wire, but employs a different geometry for the wire. Figure 6 shows a functional diagram. Here, two parallel strands of memory wire are twisted around each other; when both strands are heated they straighten, forcing the strands to become untwisted. Two such pairs of wires can provide clockwise/counterclockwise circular motions at greater torque than in the second design (since here, each individual motion is being caused by two wires straightening out rather than a single wire as in the previous design). In the Figure 6 diagram, the hinge is shown with one pair of wires in their untwisted state (on the left), and the other pair in their twisted state (at right). Subsequent heating of the wire on the right would straighten that pair while twisting the cool pair on the left. Figure 7 shows a photograph of the actuator portion of the third prototype hinge.

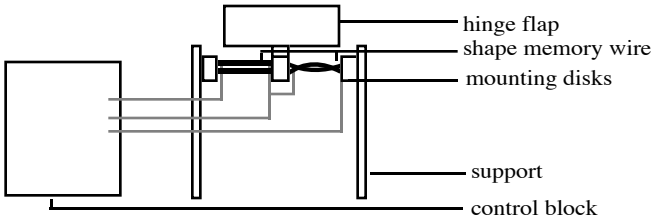

Figure 6. A diagram of the third prototype: the "twisted wire" hinge. (Note that this is a "single flap" hinge as in the first prototype.

As in the second prototype hinge, speed of response and operation is still a weak spot of this design; but the increased torque is an improvement, and the third

prototype's more compact wire geometry allows for a certain degree of flexibility in its operation. In particular, the degree of circular motion of this hinge can be changed by increasing the length of the two strands, thus increasing the number of times the pairs of wire can be twisted about each other. The relatively short range of motion of a hinge (normally no more than 180 degrees) can be provided by very short strands. In our current prototype each strand is about an inch and a half long, but less than an inch is necessary to provide 180^{o} motion.

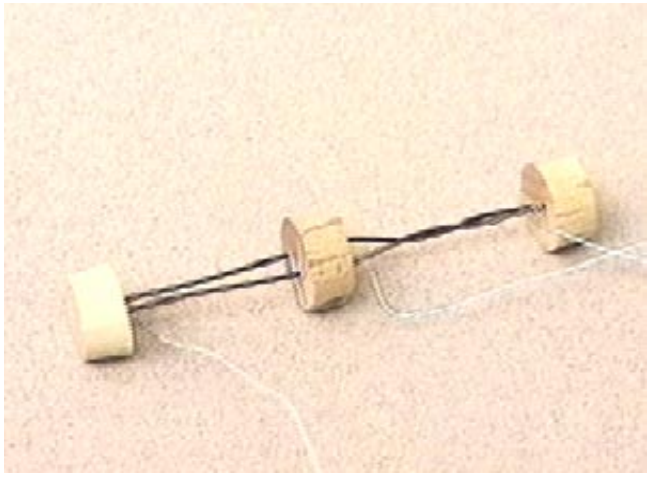

Figure 7. A photograph of the twisted shape memory wire actuator in the third prototype hinge.

Control of the hinge via a cricket.

As noted earlier, to control our various prototype hinges we used the crickets developed at the MIT Media Lab. Figure 8 shows a diagram of the cricket, which may be essentially regarded as a small independently-powered and programmed computer with 2K of memory, two sensor inputs, two actuator output ports, and primitives for communicating via infrared light with other nearby crickets.

The current version of the cricket is programmed by downloading a Logo program from a workstation running an augmented version of Microworlds Logo[S1]. This built-in Logo programming language has clear benefits: it is simple to use and contains many primitives specifically designed to control the sensor and output ports built into the cricket. Nonetheless, programming our prototype hinge proved to be a non-trivial exercise; one typical example required about 45 lines of Logo code, which to a beginner (or hobbyist more inclined to crafting than programming) could be a daunting prospect. Indeed, our experiences with crickets have prompted us to reflect on both the hardware and software issues involved in creating programmable craft elements generally; while they represent marvelous progress in that direction, the cricket and its accompanying software are still (in our view) embryonic artifacts on the path toward integrating computation and crafts. We will return to this issue later in this paper.

Figure 8. A diagram of the cricket. The two actuator output ports and sensor input ports are at the far side of the surface, toward the left. The infrared output and input are positioned along the right edge of the cricket. A standard 9V battery as power supply is housed underneath the device (which should also give the reader a sense of the object's scale).

A SAMPLE PROJECT EMPLOYING THE PROGRAMMABLE HINGE

The previous section described several models of construction for a programmable hinge; in this section we outline a possible project in which such a device might be used. Figure 9 shows a diagram of a solar power system whose purpose is to adjust itself so as to provide maximal light intensity at its solar panel.

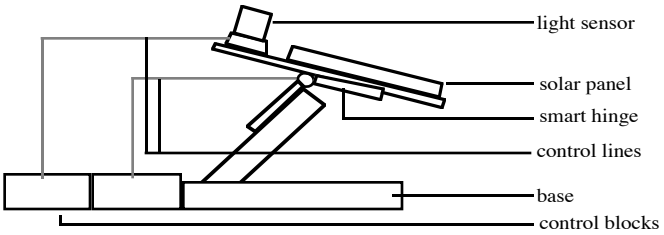

Figure 9. Diagram of a solar power system employing the programmable hinge.

The solar power system of Figure 9 employs two separate crickets (each embedded within one of the control blocks toward the left of the diagram—these would also include batteries and relays). One cricket (at left) is connected to the light sensor mounted beside the solar energy collection panel; the second cricket (at right) is part of the programmable hinge that bridges the base and collection panel. The cricket at left is able to communicate its

sensor values to its counterpart at the right via infrared signals. The light sensor beside the solar panel is partly shielded by a cylindrical sleeve, and is thus designed to read a maximal intensity value when light is impinging directly on the bottom surface of the cylinder (and on the solar panel beside the sensor as well).

The program controlling the power system is a simple hill-climbing algorithm. A sensor value is read; the hinge is adjusted slightly in one direction (say, clockwise); the sensor value is read once more; and if the new value is greater than or equal to the old, the hinge continues to adjust in the same direction as the previous move. Otherwise, if the value is less than the previous value, the hinge switches direction and adjusts counterclockwise. The result is a (slowly) adjusting panel which positions itself over time so that the light intensity is at a maximum.

This, of course, is one example of a potentially large class of projects employing even a single programmable hinge. One could imagine, for example, a programmable hinge controlling a pet door that opens only at certain times, or when the outdoor temperature is above a certain value; or a "child-proof cabinet hinge" that allows a door to be opened only when the hand gripping the door handle is greater than a certain size. The hinge might be employed at selected spots in (e.g.) marionettes or dolls, to provide algorithmic control over mouths, eyes, clapping hands, and so forth; or it could be used in an anatomical model to animate the movement of a joint. Still other possibilities will be encountered a bit later in the course of this paper.

FUNDAMENTAL ISSUES IN THE DESIGN OF COMPUTATIONALLY ENHANCED CRAFT ITEMS

While our experience with the programmable hinge is still at a very early stage, it has nonetheless given us a much wider perspective than we had previously in thinking about the integration of computers and crafts. In this section, we describe what we believe to be central issues in the design of hardware and software for new types of craft items.

Hardware Issues

Perhaps the most central hardware issue in the integration of computers and crafts is—as in so much of engineering—size (and its associated issues of weight, durability, and power). When we first began working with crickets—looking at them with the eyes of computer scientists—they seemed tiny, at 2.25 inches long and 1.25 tall and wide. After designing several rounds of programmable hinges, and thinking ahead to other types of craft items—screws, thumbtacks, string—the cricket has ballooned in our eyes to the point where it seems huge. At the scale of the hinge, one of the smaller "all-in-one" embedded controllers (like the Microchip PIC12CE[P1]) might be a more felicitous choice of hardware (though for prototyping purposes the crickets, with their associated Logo environment, have proven much easier to use); but even some of the tiny commercially available processor chips are too large for many craft items.

Similar issues arise in powering craft items. All our prototypes could be powered by a single "AA" battery (not counting the cricket itself, which is powered by a 9V battery). This is simply too large and too heavy for most applications. Moreover, the relatively short lifetime of these commercial batteries is problematic in embedded craft items; one could hardly request of users that they (e.g.) be prepared to change batteries periodically in their cabinet hinges. For these reasons, computationally enhanced craft items will necessitate creative efforts toward passive recharging (e.g., solar-powered batteries), alternative power sources, and low-power actuators.

While small computers and light portable power sources are problems of wide interest and application, there are some hardware development issues that are more specific to the design of craft items. One example—which we will return to shortly—is in the area of programming tools; for instance, one might want special-purpose hardware with which to reprogram or debug an item (such as a hinge) that is already in place within a constructed project, without the necessity of removing or replacing the item altogether. Another broad area for development is in the area of sensors and actuators. The cricket is accompanied by several standard sensor types (such as a light sensor, touch sensor, strain sensor, and so forth); but these are generally designed with an eye more toward laboratory experiments and data-gathering.

Computationally-enhanced craft items should be able to respond to the world in ways that normal craft items cannot; they therefore would be powerfully augmented by sensors specifically designed for use in craft projects. As a general rule, simplicity, low cost, and small mass are of greater importance in craft-oriented sensors than extreme accuracy: the solar power system described in the previous section, for instance, might well prove useful with a sensor that provides only a few bits of precision. Yet another possibility might be to incorporate a crude strain sensor (merely sufficient to distinguish "open" and "closed" states) within the programmable hinge; this would easily enable the hinge to employ programs that (e.g.) change behavior after a certain number of open/close cycles. Again, highly coarse-grained sound or temperature

sensors (e.g., sensors that effectively distinguish only between "above threshold" and "below threshold" for their input) could prove extremely valuable if their low precision requirements can be translated into low cost and mass.

As computationally-enhanced craft items appear in greater number and variety, they will also provide new challenges for communication. Presumably, such items will often have reason to communicate with one another: a twist of a knob might be used to signal the opening of a hinge, which itself might signal a specific pattern of tugs on a series of threads; a twist of a screw (or perhaps its loosening) at one location in a crafted artifact might be used to signal a movement at some other, moderately distant location.

In approaching the communication problem between craft items, several technologies seem worth exploring. The first is the same sort of infrared communication already supported by the crickets (which is similar to that used in commercial remote control devices). This has the advantages of low cost and easily available parts; but it requires relatively high power and a clear line of sight between the IR transceivers. A second possibility is a contact-based system—possibly based on a simple two-wire serial bus design. This technology enjoys low power requirements, low cost, and relatively high reliability, but there is a price: both a need for physical contact between craft items, and the design of a connector that will fit the needs of many different items. A third choice might be a low-power radio communication system (the transmission power would be in the microwatt range with an extremely short effective communication distance—possibly as small as one or two inches). Naturally, these three technologies need not be viewed as mutually exclusive—for instance, allowing IR transceivers to communicate via fiber optics lines might be one viable way of combining the IR and direct-contact methods for communication between items.

Software Issues

As mentioned earlier, the Logo language and environment associated with the cricket have served us extremely well in our initial experimentation with the programmable hinge; but at the same time, they affirmedly do not represent the final word in software for integrating computation and craft items. Indeed, the design philosophy behind the cricket stresses its general-purpose nature (as a small stand-alone computer); this philosophy is likewise reflected in the accompanying software, which does not assume any one particular use for the cricket and which therefore provides relatively low-level primitives for programming the device. The task of endowing craft items with computational abilities runs, in at least some ways, counter to this philosophy: a specific type of craft item (such as a hinge, nail, or string) admits of a much narrower range of realistically plausible programs than does a general-purpose computer. From the crafter's perspective, it is reasonable to assume that the effort involved in "programming a hinge", at least for typical applications, should not be substantially greater than the effort currently involved in making use of a nonprogrammable hinge.

Our belief is that it should be possible to create a suite of craft-item-specific programming primitives and environments. To take a specific example, programmable hinges would be accompanied by a "hinge-programming package", along with relatively high-level language constructs (e.g., "open-hinge", "close-hinge", "read-hinge-angle", etc.) appropriate to the particular device. Conceivably such item-specific environments might be primarily graphical in nature; while we are not, in the main, strong advocates of visual programming languages, a highly constrained set of device-specific primitives may prove to be exactly the right setting to play to the strengths of visual formalisms. Within a device-specific language environment, programs for a craft item may be kept brief and readable; in contrast, our Logo programs for the hinge have tended to be much longer and messier than their simple conceptual nature would require, primarily because of the low level constructs from which the hinge programs must be constructed.

This discussion of appropriate languages for craft items leads to still another, crucial question: what is the physical link, from the user's standpoint, between the programming environment and the craft item itself? In the case of our prototype hinges, the crickets are essentially programmed at a workstation: the user downloads her Logo program from the workstation to the cricket, which may then subsequently run the embedded program on its own. This paradigm is fine for many purposes; but, again, craft items pose their own special problems and challenges. For instance, suppose that we have programmed several hinges and placed them within a constructed artifact; and suppose that after a few tests, we find that we would like to refine or rewrite the hinge program. Ideally, we would like to be able to rewrite the program "in place", on the object itself, without disassembling the entire construction. This in turn leads to still other thoughts: perhaps, rather than programming (or reprogramming) the hinge via a workstation, we would prefer to use a much smaller device, on the order of a PDA. This would give us far greater ability to work with already-constructed devices, but it necessitates relatively simple and portable device-specific language environments—that is, language environments that can comfortably be used on a small, low-resolution screen.

Going just a bit further (and perhaps a bit more futuristically) we might consider embedding certain types of programming environments within hardware tailored for use with a particular type of craft item (e.g., one might download a simple program onto a "programmable screw" by using a "computationally enriched screwdriver"). Such speculations need not be unrealistically ambitious as long as the device-specific programming constructs are kept simple, and as long as the language environments making use of those constructs are kept minimal.

It should be noted, too, that all of this speculation is firmly in keeping with the values of the home-crafting culture. A crafted artifact, whether computationally enriched or not, should be made with stuff that can (eventually) be purchased at the local craft or hardware store and pieced together individually; it is probably a one-time effort, designed and built by a lone hobbyist of modest means within at most a few days' time. Quite possibly, the design may not have been written down at any time; and the final constructed artifact may reflect a certain degree of improvisation as well. Collectively, these are the hallmarks of a culture that is at once inexpensive, informal, creative, purposive, and suited to both the beginning and expert hobbyist; we believe that these hallmarks may be retained—perhaps even nourished—by the addition of computational media.

Before leaving the subject of software, it is worth noting still another challenge that is especially linked to the use of craft items: namely, that much of the effort in programming a crafted artifact may eventually be focused upon the interaction of many small, stand-alone programs, rather than the action of one "master" program. A home-constructed object of moderate complexity—a marionette, say, with multiple dynamic joints, or a homebuilt model of the solar system, or a kinetic sculpture—may include multiple instances of programmed craft items with novel and varied patterns of mutual communication. For such projects, it may be worthwhile to rethink the values of simplicity that we so recently advocated: perhaps yet another, higher level of programming environment is needed in this instance which will permit the user to view and simulate large collections of craft programs working in parallel. In this case, the style of the programming environment is likely to require much greater generality and flexibility; the programming languages involved are likely (in our view) to be increasingly textual (to facilitate much larger-scale and more general-purpose programs), and the site of the programmer's work is likely to return to the workstation at the desk. Whether such developments are in fact needed will be determined by the ways in which computational craft items interact: if multiple craft items offer opportunities for interesting patterns of connection (a strong likelihood, in our view), then there will eventually be a call for more powerful programming environments suited to the needs of the more expert or ambitious craftsperson.

RELATED WORK, AND CONCLUSIONS

The topics discussed in this paper spring from what has become both a venerable and burgeoning tradition of work in human-computer interfaces: namely, efforts toward the integration of the physical and virtual worlds. Such efforts are reflected in the amazingly creative work of Ishii and his colleagues in embedding computation within physical artifacts[7]; they are reflected in the development of "augmented reality" environments[6], in which computational systems may enrich the use of real-world objects (e.g., by allowing those objects to be viewed through special-purpose devices that graphically annotate or explain the objects' operations); they are reflected in the tradition of ubiquitous computing[13], in which small computational devices are spread in creative ways throughout physical environments; they are reflected in the increased interest in programming extremely large collections of simple devices in distributed media[1]; they are reflected in the growing tradition of embedding computational abilities within such objects as toys[11,12] and clothing[9].

We share with these communities an interest in blending physical and computational artifacts, and more generally in providing a counterweight to the view of computing as a relentlessly "virtual", etherealized enterprise.[4,5,14] But it is also worth restating that many of our specific concerns stem from an admiration of the culture of crafting, and those concerns are reflected (even if embryonically) in our initial development efforts. In contrast, many research projects in the area of physical/virtual integration place an emphasis on developing sophisticated, high-powered devices that cannot be programmed (or reprogrammed) by the user; in effect, these efforts reflect a longstanding engineering philosophy in which increasingly complex devices are made increasingly mysterious and opaque (though easy to use) from the standpoint of the user. Such artifacts have their place, but we prefer a style of design that emphasizes low cost, personal participation and creativity, and a certain degree of eccentric self-reliance on the part of the user. It is precisely from such communities of hobbyists and crafters that the home computer industry (and decades earlier, the radio industry) was spawned[2]; and we feel that it is important to sustain such precious communities with new, ever-more-expressive media. Perhaps, in the not-too-distant future, we will see these new media in a setting that is neither a "crafter's store" nor a "techie's

store", but something even a bit more interesting than both taken together.

ACKNOWLEDGMENTS

We are indebted to the ideas and encouragement of Hal Abelson, Robbie Berg, Fred Martin, Michael Mills, Mitchel Resnick, Brian Silverman, Jim Spohrer, and Carol Strohecker, among many others. Ann Eisenberg, Adrienne Warmack, Andee Rubin, and Vennila Ramalingam have all contributed their ideas to the work described here. Thanks to Gerhard Fischer and the members of the Center for Lifelong Learning and Design at the University of Colorado for providing an intellectual home for our efforts. This work has been supported in part by the National Science Foundation and the Advanced Research Projects Agency under Cooperative Agreement CDA-9408607, and by NSF grants CDA-9616444 and REC-961396. The second author is also supported by an NSF Young Investigator award IRI-9258684. Finally, we would like to thank Apple Computer, Inc. for donating the machines with which our research was conducted.

REFERENCES

1. Berlin, A. and Gabriel, K. [1997] Distributed MEMS: New Challenges for Computation.*IEEE Computational Science and Engineering*, Jan-March 1997, pp. 12-16.
2. Campbell-Kelly, M. and Aspray, W. [1997] *Computer: a History of the Information Machine*. Harper Collins.
3. Druin, A. [1987] Building an alternative to the traditional computer terminal. Master's Thesis: MIT Media Lab.
4. Eisenberg, M. and Eisenberg, A. N. [1998] Middle Tech: Blurring the Division between High and Low Tech in Education. To appear in Druin, ed. *The Design of Children's Technology*. Morgan Kaufmann.
5. Eisenberg, M.; Mackay, W.; Druin, A.; Lehman, S.; and Resnick, M. [1996] Real Meets Virtual. Panel Session, *CHI '96 Conference Companion*, pp. 157-158.
6. Feiner, S.; MacIntyre, B.; and Seligmann, D. [1993]Knowledge-based Augmented Reality. *Communications of the ACM*, 36:7, pp. 52-62.
7. Gilbertson, R. [1993] *Muscle Wires Project Book*. Mondo-Tronics.
8. Ishii, H. and Ullmer, B. [1997] Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms. *Proceedings of CHI '97*, pp. 234-241.
9. Mann, S. [1997] Wearable Computing: a First Step Toward Personal Imaging. *IEEE Computer*, 30:2, pp. 25-32.
10. Resnick, M. [1993] Behavior construction kits. *Communications of the ACM*, 36:7, pp. 64-71.
11. Resnick, M.; Martin, F.; Berg, R.; Borovoy, R.; Colella, V.; Kramer, K.; and Silverman, B. [1998] Digital Manipulatives: New Toys to Think With. *Proceedings of CHI '98*, pp. 281-287.
12. Umaschi, M. [1997] Soft Toys with Computer Hearts. *Proceedings of CHI '97*, Atlanta, pp. 20-21.
13. Weiser, M. [1993] "Some Computer Science Problems in Ubiquitous Computing," *Communications of the ACM*, July 1993.
14. Wellner, P.; Mackay, W.; and Gold, R. [1993] Computer Augmented Environments: Back to the Real World. *Communications of the ACM*, 36:7, pp. 24-226.

Software

[S1] Logo Computer Systems, Inc. (http://www.lcsi.ca/)

Products

[P1] Microchip Technology Inc. (http://www.microchip.com/)