Knowledge-Based Systems 13 (2000) 471–478
www.elsevier.com/locate/knosys

Integrating craft materials and computation☆

G. Blauvelt*, T. Wrensch, M. Eisenberg

Department of Computer Science, Institute of Cognitive Science, and Center for Lifelong Learning and Design, University of Colorado, Boulder, CO 80309-0430, USA

Abstract

Traditionally, the notion of home crafting connotes the use of "low-tech" materials and techniques; but increasingly, the once-distinct worlds of crafting and computational media have become integrated, to the mutual benefit of both cultures. In this paper, we discuss a wide range of recurring issues in the integration of crafts and computation, drawing upon a variety of related research projects. In particular, we explore the ways in which attention to computational crafts can encourage a productive re-examination of such notions as programming languages, computer architectures, and peripheral devices. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Computation and crafts; HyperGami; HyperSpider; computationally enriched craft items

1. Introduction

Traditionally, the very notion of home crafts carries connotations of a world of "low-tech" materials—paper, clay, wood, fabric, and so forth. But increasingly, the once-distinct worlds of home crafting and computational media have become integrated on a variety of fronts. Software applications may be used as instructional or design media for craft activities (such as origami[1] or quilting[2]); and computational capabilities may be incorporated into materials themselves, as in the development of "wearable computing" [11]. This continual and creative erosion of the boundary between crafting and computation (or alternatively, between "low" and "high" technology) has important ramifications for both cultures. On the craft side, the advent of computation both enhances the expressive capabilities of existing materials, and supports the development of "intelligent" materials that form the basis of entirely new branches of craftwork. Perhaps less obviously, on the computational side, a focus on craft activities can lead to a re-evaluation of some of the basic concepts of traditional computer science: a "computational crafter" may well begin to rethink the very ideas of programming languages, software engineering, computer architecture, and peripheral devices.

This paper is an exploration and survey of our own still-nascent ideas about the integration of computation and crafts, with particular emphasis on how the use of inexpensive, home-accessible physical media can impact the design of computer hardware and software. The examples that motivate this paper have emerged from our own work, and the work of our colleagues at the University of Colorado, over the past several years. Some of these examples are in fact fully working programs; some are in the nature of working prototypes; some are still works-in-progress; and a couple are frankly speculative. Collectively, however, these efforts share a focus on bridging the long-standing gap between the "abstract" world of programming and home computing on the one hand, and the "tangible" world of crafting materials on the other for the mutual enrichment of both.

The remainder of this paper is organized as follows. In the second section, we describe several major strategies—design paradigms—for blending crafts and computation. In presenting these strategies, we concurrently describe various relevant particular projects that we and our colleagues have undertaken, illustrating the strategies under discussion. The remaining five sections of the paper use these projects as "objects to think with" in order to explore potentially important new issues in computational crafting. The five sections explore, in turn, issues in programming language design; the creation of computationally enhanced craft objects; the design of novel computer peripherals; the

* Corresponding author. Tel.: +1-303-492-8091.
E-mail addresses: zathras@cs.colorado.edu (G. Blauvelt), wrensch@cs.colorado.edu (T. Wrensch), duck@cs.colorado.edu (M. Eisenberg).

[1] *Origami: The Secret Life of Paper* (CD-ROM.) Cloudrunner, Inc. Published by Casady & Greene, Salinas, CA.

[2] *Electric Quilt 4.0* Patchworks Studio, Sidney, BC, Canada.

☆ Portions from the Proceedings of the Third Conference on Creativity and Cognition, Loughborough, UK, October 10–13, 1999, pp. 50–56. Reproduced with permission from ACM © 1999.

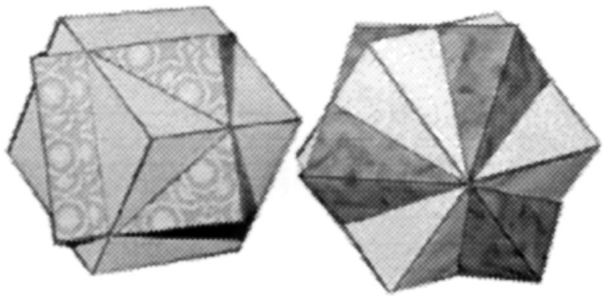


Fig. 1. Two compound polyhedra (each is a compound of two cubes) constructed with HyperGami.

design of programming "conduits" (i.e. devices for transferring programs to physical media); and the design of large-scale computational craft systems. While most of the ideas presented in these sections are at a very early stage of formation, they have emerged only slowly from a growing body of experience (most of it still clumsy and inexpert) in computational crafting.

2. Strategies for blending crafts and computation

While there are undoubtedly many creative ways of integrating craft materials and computation, in our own experience a surprisingly large variety of specific projects exhibit one of three major categories of design strategy. In this section, we describe these three design strategies and (very briefly) present those projects that serve to illustrate them.

2.1. Strategy 1: software applications for "traditional" crafting

Probably the most straightforward design strategy for integrating computation and crafts is to regard the computer as a tool with which to design craft objects. In this model, a stand-alone computer application aids the practitioner in the creation of craft projects. Before elaborating on this model further, we present two sample projects by way of illustration:

- HyperGami [6] is a computer application (written in the Scheme dialect of Lisp) for the design of customized paper polyhedral models and sculptures. The basic idea behind the application is that it allows the user to create novel three-dimensional shapes on the computer screen; the software then "unfolds" those shapes into two-dimensional "folding net" patterns, which may then be decorated by a variety of means. The folding net pattern is output to a color printer, and the user completes the project by building a tangible three-dimensional paper model. HyperGami and its more recent Java-based descendant JavaGami [4] have been used in numerous tutorials, project-based classes, and workshops over the past several years; and an extensive variety of polyhedra and sculptures have been created with the system. Two examples are shown in Fig. 1, and many more may be seen in the sources referenced.
- HyperSpider [7] is a Java-based computer application designed by T. Chen for the creation of mathematical string sculptures. In this program (much as in HyperGami), the user designs a three-dimensional representation of a particular string pattern on the computer screen (here, the string consists of a series of colored threads strung between three mutually orthogonal planes); the program then displays a "flattened" representation of the pattern that facilitates the construction of the pattern with thread and cardboard. HyperSpider, while a working application, is a much more recent effort than HyperGami; but a sample project made with the assistance of the program may be seen in Fig. 2.

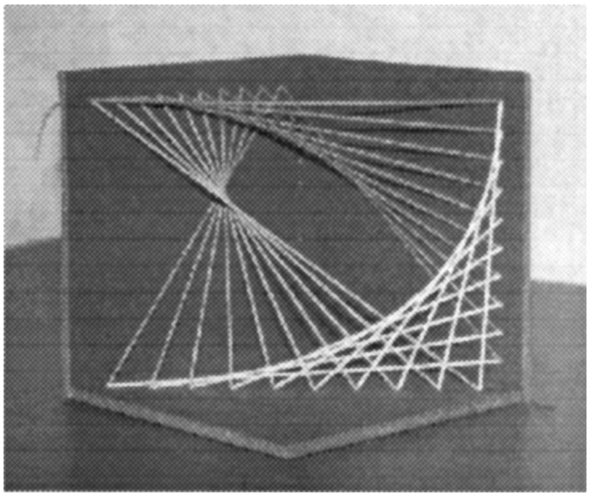


Fig. 2. A string sculpture designed in HyperSpider.

In both HyperGami and HyperSpider, a computer application is used to design objects that are then created away from the computer screen. Moreover, the materials in question—here, paper and string—are "traditional" craft materials whose historical uses were established well before the era of computation. It is worth noting, however, that even in these traditional arenas, computer applications may be used to design more complex objects than are typically feasible otherwise; they may be profitably combined with a wide range of external computational systems (e.g. the World Wide Web may be used for the display and sharing of new designs); and, most importantly, they act as platforms through which new notations and formalisms for craft activities may be explored. We will return to this last issue in the next section; and for more extended discussion of all these points, see [7].

2.2. Strategy 2: computationally enhanced craft items

In the first design strategy described above, computer applications are used to create craft items built from

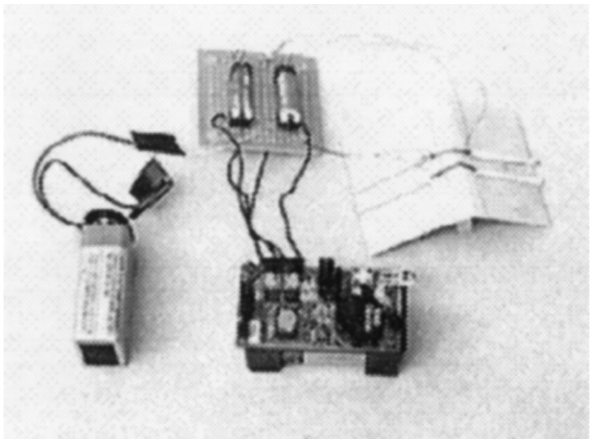


Fig. 3. A photograph of a working prototype of the programmable hinge. The hinge's battery power source is visible toward the left of the photo; the associated computer (a Media Lab cricket) storing the hinge's program is at bottom center; and the hinge itself, with shape memory wire actuator, is seen toward the right.

traditional (or at least "noncomputational") materials. The past decade, however, has seen a vastly accelerated enterprise of blending computation into materials themselves, in the form of embedded computation, ubiquitous computing, "smart" materials, and so forth. (See, for instance, Gershenfeld [9] for a highly readable account of some of the most interesting examples of this work.) The increasing feasibility of weaving computational capabilities into materials has provocative implications for computational crafting. In our own work, we have explored several avenues by which various forms of small, commonly encountered craft items may be endowed with programmable behaviors. Brief descriptions of several of these "computationally enhanced craft items" follow.

- The programmable hinge [17] is a craft item modeled on the type of hinge that might be found in a kitchen cabinet or jewelry box lid. The working prototype shown in Fig. 3 employs one of the MIT Media Lab's "crickets" (or programmable Lego bricks[14]) to provide the computation for the device, and shape memory alloy actuators to open and close the hinge. While the prototype is still rather unwieldy the computational element here is not sufficiently integrated within the craft object itself, the result is nonetheless a hinge-like object that can be programmed with simple behaviors such as "open and close five times; wait one minute; then repeat indefinitely".
- The "rototack" [16] is a craft item based on a thumbtack. While standard thumbtacks are rather static objects, with little behavior to automate, the rototack is able to rotate its head around its axis. Thus the tack is a source of controlled circular motion that could be used, for example to drive a set of mechanical components such as gears, cams, and pulleys. Using a standard cork board as a work area, rototacks could form the basis of a rapid, expressive way to experiment with a variety of mechanical motions. Fig. 4 shows one of our first prototypes of the tack.

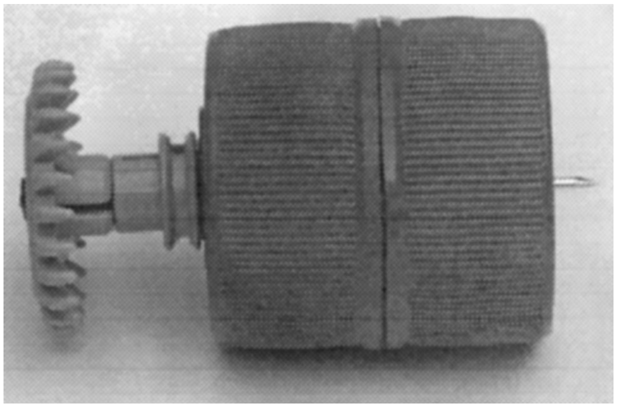


Fig. 4. A side view of an early rototack. The "pin" may be seen at right; the rotating head at left. The power source and internal computer is hidden within the casing at center.

- The programmable ceramic tile, a craft item for which we have a still-very-early prototype, is based on the observation that ceramic tiles have a variety of useful and interesting properties: they are waterproof, resistant to extremes of temperature, can be made in a variety of shapes, sizes, and colors, and can be used to tile surfaces of arbitrary shape (such as cylinders, vases, and globes). Programmable tiles add the ability to sense when they are touched; to light up or change their color in response to this input; and to communicate with other adjoining programmable tiles. Groups of such tiles could be used to create active tile mosaics, add interactivity to the exteriors of buildings, or add intelligence to a shower stall.

2.3. Strategy 3: heterogeneous craft programming, and simulation environments

Perhaps the most ambitious marriage of computation and crafts is to build systems in which: (a) a variety of computational and noncomputational materials may be used in craft projects; (b) at least some of these materials are designed to facilitate inter-object communication; and (c) the accompanying software environment may be used to program individual objects (as in the case of the examples above), to simulate larger systems of those objects, or to communicate directly with craft objects. The first of these points derives from the fact that craft projects not atypically include a wide range of materials, and that there is no reason to exclude the possibility of more than one computationally enriched craft element occurring within any given project. The second point derives from the fact that there are many potential ways—mechanical, electrical, optical, chemical—by which craft objects can in principle communicate with one another, and computationally enriched craft objects may well be designed with any of these modes of communication in mind. The third point derives from the fact that software applications may eventually be used to design (and communicate with) objects whose parts consist of a collection of craft materials, some of which may themselves have computational capabilities: in effect, this is looking toward an elaboration of the notion of computer-aided design to include home projects employing both low- and high-tech materials.

While we do not have any one particular project that exemplifies the full (and ambitious) range of this design strategy, we have recently begun work toward creating an application for the design of homemade automata (of the sort exemplified by the remarkable artists of the Cabaret Mechanical Theatre museum in London [13]). The mechanical ingredients of these automata include elements such as gears, ratchets, cams, and linkages, all employed in the service of creating a playfully dynamic work of art. An application of the sort that we envision would ideally include design units for the individual mechanical elements; means by which combinations of those elements could be simulated in concert; browsable libraries and databases of sample automata; and software that would describe techniques for constructing the desired mechanical elements from a wide variety of easily obtained materials (perhaps providing cut-out patterns in the manner of HyperGami as well). Just to take an example: a student who wishes to build an automaton of a dog wagging its tail could begin by looking through a collection of automata either with similar subject matter (animals) or perhaps similar types of movement (oscillatory patterns like those of the dog's tail). Using these examples as starting points, she could then custom-design her own mechanical elements (such as cams) on the screen and simulate at least a simple schematic of her newly created elements. Finally, she could request that the application either provide a printable pattern for her mechanical designs (to be rendered in, say, cardstock) or to provide advice and guidelines for constructing these elements from other materials.

The idea is still at an early formulative stage, but the domain of automaton design is an appropriate one in which to explore the more ambitious directions of a "system-building" design strategy. First, an automaton application should be used both to design the components of an object and to simulate the resulting object. Second, a variety of heterogeneous craft materials are likely to be used in the construction of real-world automata-wood, metal, paper, string, and plastic are among the typical materials used. And finally, the domain affords scope for exploration into more complex realms of construction (e.g. one might imagine ultimately including programmable craft items like the rototack within our machines).

3. Crafts and computers: language design

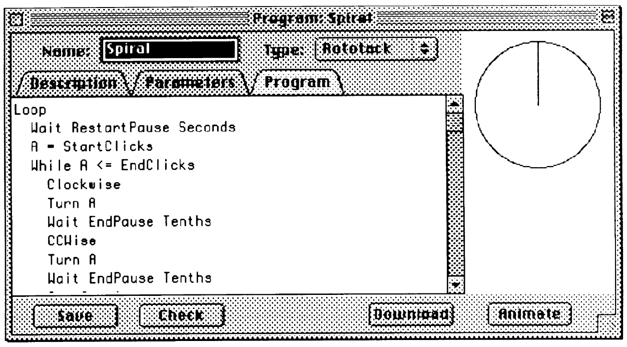
The previous section described several broad strategies for the integration of crafts and computation. In this and the remaining sections, we explore the ways in which these strategies (and the various projects that exemplify them) may be used as springboards for re-evaluating traditional concepts in computer science.

3.1. Devising new languages for craft domains

The first such concept is the notion of a programming language. Many home crafts have long been associated with "linguistic" notations, often informal or implicit—a glance at any origami book will reveal a sort of rough-and-ready graphical notation for folding, while knitting patterns have their own traditional notation (see Nardi [12] for an illuminating discussion of this latter example). One powerful influence of computational media upon crafting cultures is precisely along this linguistic dimension notations can be brought to the forefront, and made an object of exploration in their own right. Thus, in the HyperGami and HyperSpider projects, we have used the crafts of polyhedral modelling and mathematical string sculpture, respectively, as domains in which to explore new computational "algebras" for the creation of physical objects. A HyperGami designer, for instance, applies a variety of functions (such as "stretching", "capping", "slicing", and so forth) to classical shapes in order to create new polyhedra for the system to unfold; and importantly, these functions may be applied in combination, so that new polyhedra may themselves act as the starting points for still other customized forms. In a similar fashion, one could imagine inventing new and powerful notations for traditional crafts like knitting; but perhaps a still more exciting prospect is to imagine devising computational algebras for a variety of crafts that currently have only rudimentary notations (such as pop-up design, sliding-tile puzzle design, shadow-puppet design, and kaleidoscope design, just to name a few). The essential point here is that notations are not merely used to record craft activities, but to provide an underlying conceptual structure for those activities: that is, they act as media that suggest directions for development and experimentation. Computational media, then, can spur crafters toward the creation of crafting languages which themselves act as the means by which entirely new vistas of craft technique can be discovered and expressed [7].

3.2. Geometry-specific Languages

The development of computationally enhanced craft items (such as the programmable hinge and tack described earlier) suggests yet another direction for exploring the idea of a "programming language". Traditionally, higher-level computer languages have been deliberately devised so as not to reflect the physical device in which their programs will be run; indeed, this is historically the motivating reason behind the creation of higher-level languages. Thus—just to illustrate—the Basic language is an artifact that can (at least to a very large extent) be studied independent of the



```
Loop
  Wait RestartPause Seconds
  A = StartClicks
  While A <= EndClicks
    Clockwise
    Turn A
    Wait EndPause Tenths
    CCWise
    Turn A
    Wait EndPause Tenths
```

Fig. 5. A mockup screen of a rototack programming environment. As envisioned here, the program is written and edited in the text window at left; by pressing the Animate button at bottom right, the program will be interpreted and, while it runs, the graphical schematic of the tack will show the rotation that a physical tack would exhibit. Once the user is satisfied with the behavior of the program, he can download it to an actual tack.

particular choice of machine on which the language will be run. But this imposition of a conceptual divide between hardware and software so reasonable when the hardware in question is a general-purpose computer may be far less defensible when the hardware is a small craft item with a specific geometry and limited computational resources. Thus, devising a "computational tack language" means, in effect, creating a notation for actions taken by an object that can do no more than rotate; a "hinge language" is a notation for small objects that bend around a central axis. In both these cases, our desire is to create highly simple languages by traditional computer science standards they may include little, for instance, in the way of available data or control structures but that are nonetheless rich in their expressive potential for objects with certain stereotyped forms of physical action. We could likewise imagine other languages created around particular modes of physical movement—a "tensile" language for strings, for instance, or a "rolling" language for controllable spherical objects.

The design of such "geometry-specific" (or perhaps "behavior-specific") languages would naturally be accompanied by a re-examination of the systems that support languages as well: editors, debuggers, program browsers and libraries, and so forth. To take a natural example: a language environment specifically created for our hypothetical "rototack language" might well include animation tools with which to depict, in schematic form, the rotation of a tack as a particular program runs. Fig. 5 shows an illustration of the idea: here, a mockup screen of a rototack programming environment includes a text window containing a rototack program editor (this employs a simple imperative-language style syntax), as well as a graphical window in which the rotation of the tack can be shown as the program runs. Thus, when the user tests the program with a rototack language interpreter, the animation at right shows how a physical tack containing this program would behave. In a similar vein, we could envision a program browser in which the rototack programmer surveys a series of representative tack animations, selecting the behavior closest to the one he wishes to generate; by selecting the appropriate example, the programmer would then have a starting sample program that could be edited and customized for the programmer's particular ends.

Neither of the scenarios mentioned in the previous paragraph, for a program animation or a graphical program library, is particularly novel; indeed, proponents of algorithm animation [2,3] have long advocated (and designed) just such systems. The crucial point here, however, is that craft-object language environments provide an ideal small-scale venue in which to embed these ideas into working end-user-programming environments. The behaviors of (e.g.) programmable tacks and hinges are sufficiently simple and specific so that they provide a feasible setting in which to explore long-established experimental techniques (such as algorithm animation) with tiny programs and with populations of nonprofessional users [5].

4. Crafts and computers: craft object design

4.1. Computing below the turing machine level

Perhaps the most influential theoretical model in computer science is that of the Turing machine, first described in Alan Turing's classic 1936 paper [15]. One of the remarkable features of this model is its simplicity: a Turing machine consists merely of an infinite linear array of discrete memory cells, and a finite-state machine with a read/write mechanism that can read characters from, and write characters to, those cells while moving along the array one cell at a step in either direction. While real-world computers are of course much more elaborate than this, the essential lesson from Turing's model is that it does not take much machinery to achieve the status of a complete, bona fide computer (at least if one allows a "very large" finite memory to act as a real-world approximation to an infinite tape).

Much of practical computer science since Turing's time has been taken up with building devices that are ever-more-powerful variants of the Turing machine: faster memories, faster processors, multiple processors, and so forth. What is striking, in contrast, about the integration of computational media and crafts is that it might well compel us to think once more about computational devices below the threshold of Turing machine computation. Specifically, a computationally enriched hinge or tack might well be assumed to have only a very small internal store of read/write memory, on the order of a few bits; if anything, our own early prototypes of these devices have much more memory than one would want in a smaller, more functional device. To put the matter another way: computationally enriched craft items are likely to come in a variety of levels of sophistication, ranging from larger items wherein factors such as cost, size, and flexibility have been sacrificed for computational power, to smaller items in which the tradeoffs have been made in the other direction. This being the case, we should now look to the idea of "sub-computers": computational devices that: (a) are perhaps a bit too large to treat in practical terms as simple finite-state devices; (b) are conversely too small to treat as fully fledged computers; and (c) whose complexity arises from the behavior of collections of the devices (e.g. from multiple-hinge projects) rather than from the computational power of any single device.

4.2. Craft items as "partial" or "fragmented" computers

A continuation of the preceding discussion leads us to the idea that computationally enhanced craft items might well be considered as "sub-computers" both on the architectural and theoretical levels. Traditionally, what we think of as a "computer" has all the necessary architectural ingredients bundled into one case-processor, memory, power connection, and so forth. The advent of small, flexible computational craft items, however, suggests that we explore splitting off the functionality of traditional computers into distinct portions so as to achieve greater functionality. For instance, we are currently considering designs in which computational tacks are placed in computationally enriched surfaces (perhaps these might be called "computational cork boards"). In this design, the cork-board might house a power source and processor: the computational role of the tack might now be merely to hold a single routine that is called by the processor within the board itself.

Space limitations preclude a much more extended discussion of this idea, but the fundamental direction should nonetheless be emphasized: it might well be feasible to think of computationally enriched craft items as "fragments" of full computers-small chunks of memory attached to a subroutine that only achieve their power, and take physical action, in the context of larger background devices that supply the remainder of the necessary computational architecture.

5. Crafts and computers: peripherals

5.1. The role of the printer

In the course of our experiences with the HyperGami system, we have found ourselves exploring the ever-expanding range of materials available for use with printers: transparency material, cardstock, glossy specialty papers, and so forth. HyperGami nets have been printed on special material that may then be transferred to fabric: this technique is usually employed to make customized T-shirts, but we have used it to construct (stuffed) polyhedra out of sewn cloth.[6] In addition, there are numerous specialty materials that can be used to augment constructions after they have emerged from a printer: flexible polarized filters, papers with holographic designs, glow-in-the-dark paints, and many more.

The upshot of all this exploration is that, by combining computational and craft media, we can vastly expand our traditional notions of the power of computer peripherals such as printers. Rather than simply printing out paper, a "printer" may be better conceived of as an output device for a larger variety of flexible decorable sheet-like materials. Indeed, Gershenfeld [9] describes a variety of architectures for "3D printers" whose output is in the form of a tangible object (made from a material such as plastic that has been cured by printer-controlled ultraviolet light). While such devices are presently either at the research stage or are beyond the resources of the home hobbyist, it is not at all unrealistic to explore the potential of these ideas for home crafting.

Moreover, we need not approach this subject in quite so futuristic a vein. There are somewhat less exotic (and presumably more affordable) specialized printers that could be of particular value to the computational crafter. In our own work, for example, we have made recent use of a computer-controlled laser cutter that can (e.g.) cut thin slabs of wood to produce objects of customized shapes. This device is especially handy for creating the sorts of user-created gears and cams that could go into the construction of automata like those mentioned earlier; that is, the materials appropriate to the laser cutter (such as basswood) can be used to create sturdy linkages much more readily than traditional "printer" materials such as cardstock. The point of this example is merely to suggest that the landscape of printing devices, viewed more broadly as customized material-output devices, is already poised for a rapid expansion.

And finally, it should be added that we need not view printers only as instances of output devices; for the purposes of the home crafter, we might alternatively focus on the role of the traditional printer as a "decorating" device for transferring designs to surfaces. In this case, our speculations would be in the direction of devising printers for a variety of non-sheet-like surfaces (e.g. "printers" for applying decorations to spherical objects).

5.2. Craft-compatible readers, sensors, actuators, and monitors

Taking courage from the previous paragraphs, it is perhaps only a bit more futuristic to imagine a "wish list" of peripheral devices other than printers. One could imagine a role for scanning devices other than the usual character- or barcode readers: for instance, a simple portable device that could "read" a color value from a surface might well find its way into a variety of craft projects. "Readers" for still other material qualities-texture, viscosity, conductance-might likewise find a variety of uses in crafting projects.

Indeed, "computer peripherals", for the purposes of crafting projects, might profitably be thought of in terms of sensors and actuators. For example, one might design a computer-controlled actuator whose role is simply to issue a discrete tug along a string, or to issue a brief tap upon a surface. (The former actuator might find a role in, say, computationally controlled marionettes; the latter in a computationally controlled ripple tank.) Such devices could certainly be engineered given the current level of technology, but they are not yet at the level of commonplace, inexpensive computer accessories. Still another direction for exploration-this one representing a greater engineering challenge-would be the extension of the notion of "computer monitor" to a variety of arbitrary surfaces. One might well wish to augment all sorts of objects or geometric shapes with (even rudimentary) displays: a metal wire might, for instance, be augmented with a small, inexpensive display indicating (say, via color) the level of current going through it.

6. Crafts and computers: program transfer

In traditional computer science, the process by which a program is communicated to a computer's memory either from the text written by a programmer or from an externally stored copy of the program is usually so immediate (or uninteresting) as to escape notice: one simply types a program into a keyboard, or reads the program from secondary memory, and the rest of the process is invisible. But in the realm of computational crafts, where "computers" may be sitting in inaccessible locations (say, behind ceramic tiles), the process of communicating a program to a computer suddenly becomes a problem worthy of study. More generally, the problem of communicating a program to a computationally enhanced craft item (or reading that item's current program) while not displacing or disturbing that item is an issue that calls for a variety of inventive solutions.

Just to illustrate how the problem is dealt with in one particularly interesting case: the MIT Media Lab's "crickets", for example, receive their programs via infrared light. The user first writes a program in a special dialect of Logo on a desktop computer; then the "cricket" is brought within range of an IR signal sent to it from an attachment to that computer. The overall process is one by which the user can write a program in a powerful programming environment on the desktop, but can then transfer that program to the small mobile computer.

6.1. *"Wands" and other geometry-specific program conduits*

While the program-transfer strategy employed by the cricket is clever and useful, it unfortunately does not apply to all potential cases in the broader realm of computational crafts. For example, as mentioned before, the "computer" to be programmed (or re-programmed) may be sitting in an inaccessible location, or may be immobile. In this instance, rather than bring the embedded computer to the program, we need to somehow bring the program to the computer.

During the past year we have begun exploring one style of solution to this problem: a programming "wand" that can be held in the hand, and whose function is simply to transfer a program to an embedded computer via low-power radio waves or IR light. The wand, as planned, should be shaped as a long extensible rod that enables the user to reach into otherwise inaccessible spots (e.g. behind a large immobile object, or up toward a ceiling tile) from which point it can communicate its program to an embedded device.

The programming wand is really just one instance of what might be a larger class of geometry-specific program "conduits" whose task is to transfer user-specified behavior to craft objects. Just as the craft objects themselves may have unusual or idiosyncratic geometries, so might the means for programming those objects. Conceivably, program "conduits" might take the form of long flexible wires, or thin sheets; or mobile devices might be programmed to travel into inaccessible spots and "deliver" programs to other devices.

6.2. *Programming a system en masse*

Before leaving this topic, it is worth mentioning still another variant of the program-transfer task that arises in the domain of computational crafts. Consider a situation, for example, in which we have a large array of programmable ceramic tiles embedded within a flat surface. Quite possibly, we might wish to transfer a particular program (say, the code for playing Conway's game of Life [8]) to each and every one of these tiles. In such a case, reprogramming each tile individually would be unwieldy: that is, we would not use to reprogram each of the tiles with the "wand" described in the previous paragraphs. The issue in this case is how to program a (potentially large) set of objects all at once. In this instance, using a communication device such as sound might be feasible; or we might need to arrange additional techniques by which craft items can themselves act as program-transfer devices, communicating new programs to each other. (In the case of a rectangular array of programmable ceramic tiles, this notion seems reasonable in that we might arrange for each tile to communicate a new program to its four immediate neighbors.)

7. Crafts and computers: systems

Perhaps the greatest challenges in blending computational media and crafts involve the myriad ways in which new craft objects might be expected to communicate with one another and with more traditional desktop computers. In this final section, we explore these larger "systems" issues, looking toward the longer-term future of computational crafts.

7.1. *Coherent systems of craft elements*

Many existing craft domains involve kits of dedicated domain-specific pieces that mesh together to produce larger structures: plastic anatomical models, geometric building blocks, train track pieces, and so forth. A general systems-level problem in computational crafting, then, would be to devise techniques for inter-object communication within such domain-specific kits. Currently, some tantalizing examples of computational construction kits exist [1,10], but even these experimental systems are relatively general-purpose in structure. In contrast, more craft-specific construction kits could incorporate software tailored to the particular domains in question. A collection of computationally enhanced architectural building blocks might communicate with one another so that the overall construction "knows" something about the stability of the structure created; a snap-together polyhedral construction might be able to communicate its overall symmetry to the user; a train-track pattern might "know" whether it is topologically equivalent to a figure-eight, or how many crossings it contains. In other words, there are interesting problems to be solved in endowing craft kits with both general protocols for communication and domain-specific knowledge to make use of the information thus communicated.

7.2. Communications protocols between heterogeneous objects

The previous paragraph dealt with communication within systematic "kits" of similar objects, such as train-track segments. More generally, though, the computational crafter will eventually want standardized means by which, e.g. craft objects can communicate with desktop computers, the World Wide Web, or even other, unrelated craft objects. One might want to devise projects in which (for instance) programmable tiles can communicate a signal not only to other tiles but to some other object (such as a hinge); or one might want information obtained from the Web to influence the behavior of a building's exterior wall of tiles. Such general inter-object communication may well be the ultimate goal of the computational crafter; but it may yet be premature to worry about devising overarching standards for heterogeneous craft objects. Our own view is that a great deal more "lore", in the form of anarchic smaller-scale projects, will first need to be explored before we can productively embark on the creation of a larger-scale infrastructure of computational crafting.

Acknowledgements

Ann Nishioka Eisenberg, Ted Chen, Andee Rubin, and Vennila Ramalingam have all contributed to the efforts described in this chapter. We are indebted to the ideas and conversation of Robbie Berg, Gerhard Fischer, Fred Martin, Mitchel Resnick, Brian Silverman, and Carol Strohecker, among many others. The first author is supported by a gift from the Mitsubishi Electric Research Laboratories (MERL) in Cambridge, Massachusetts. The work described in this paper has also been supported in part by National Science Foundation grants CDA-9616444 and REC-961396; by the NSF and the Advanced Research Projects Agency under Cooperative Agreement CDA-9408607; and by a Young Investigator award IRI-9258684 to the third author. Finally, we would like to thank Apple Computer, Inc. for donating the machines with which this research was conducted.

References

[1] D. Anderson, J. Frankel, J. Marks, D. Leigh, K. Ryall, E. Sullivan, J. Yedidia, Building virtual structures with physical blocks (Demo Summary). Proceedings of UIST 99, Asheville, North Carolina, USA, 1999.
[2] M. Brown, Exploring algorithms using Balsa-II, IEEE Computer, May (1988) 14–36.
[3] M. Brown, R. Sedgewick, Techniques for algorithm animation, IEEE Software, January (1985) 28–39.
[4] A. Eisenberg, An educational program for paper sculpture, PhD thesis, University of Colorado Computer Science Department, 1999.
[5] M. Eisenberg, T. Wrensch, G. Blauvelt, Geometry-specific languages, Colorado Department of Computer Science Technical Report CU-CS-886-99, October 1999.
[6] M. Eisenberg, A. Eisenberg, Middle tech: blurring the division between high and low tech in education, in: A. Druin (Ed.), The Design of Children's Technology, Morgan Kaufmann, San Francisco, 1999, pp. 244–273.
[7] M. Eisenberg, A. Rubin, T. Chen, Computation and educational handicrafts: a strategy for integrative design, Proceedings of International Conference on the Learning Sciences, Atlanta, December 1998.
[8] M. Gardner, Wheels, Life, and Other Mathematical Amusements, New York, Freeman, 1900.
[9] N. Gershenfeld, When Things Begin to Think, Holt, New York, 1999.
[10] M.G. Gorbet, M. Orth, H. Ishii, Triangles: tangible interface for manipulation and exploration of digital information topography. Proceedings of CHI 98, Los Angeles, CA, 1998, pp. 49–56.
[11] S. Mann, Wearable computing: a first step toward personal imaging, IEEE Computer 30 (2) (1997) 25–32.
[12] B. Nardi, A Small Matter of Programming, MIT Press, Cambridge, MA, 1993.
[13] A.L. Onn, G. Alexander, Cabaret mechanical movement, Cabaret Mechanical Theatre, London, 1998.
[14] M. Resnick, F. Martin, R. Sargent, B. Silverman, Programmable bricks: toys to think with, IBM Syst. J. 35 (3) (1996) 443–452.
[15] A. Turing, On Computable Numbers, with an application to the Entscheidungsproblem, Proc. Lond. Math. Soc. (2) (1936) 230–265.
[16] T. Wrensch, G. Blauvelt, M. Eisenberg, The Rototack: designing a computationally-enriched craft item. To appear in the Proceedings of DARE (Designing Augmented Reality Environments), Elsinore, Denmark. 2000.
[17] T. Wrensch, M. Eisenberg, The programmable hinge: toward computationally enhanced crafts, Proceedings of UIST 98, San Francisco, November, 1998, pp. 89–96.