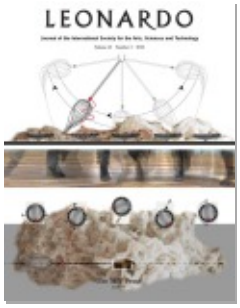
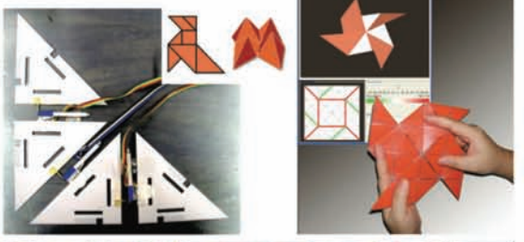
Educating the New Makers: Cross-Disciplinary Creativity

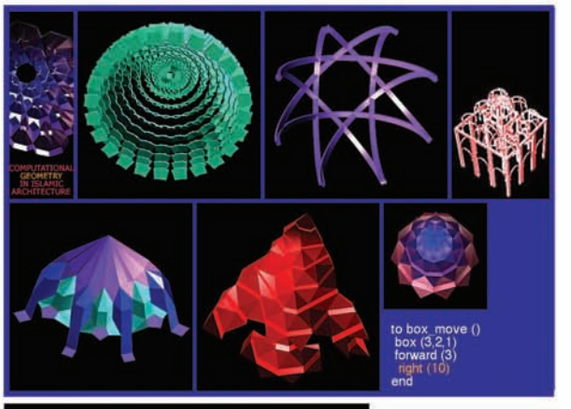
Mark D. Gross
Ellen Yi-Luen Do

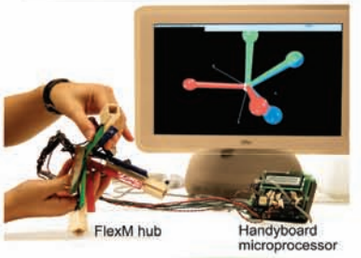
Leonardo, Volume 42, Number 3, June 2009, pp. 210-215 (Article)

Published by The MIT Press

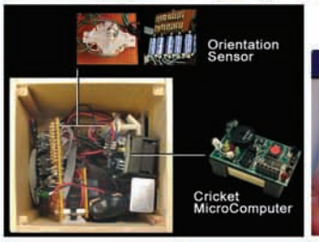


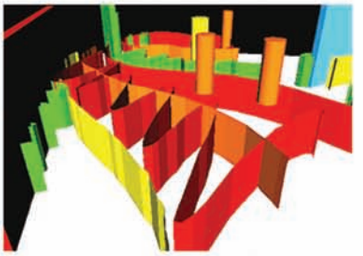


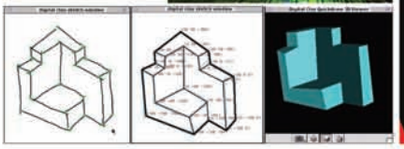

COMPUTATIONAL GEOMETRY IN ISLAMIC ARCHITECTURE
to box_move ()
box (3,2,1)
forward (3)
right (10)
end

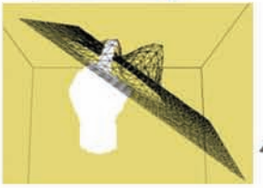

FlexM hub
Handyboard microprocessor

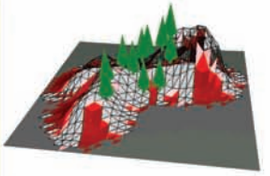

configure
window
motor
— bench

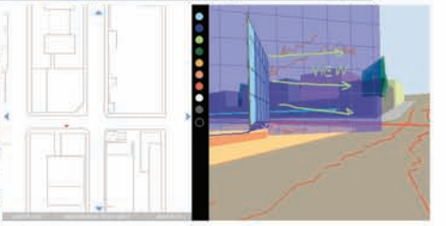

Orientation Sensor
Cricket MicroComputer











SPECIAL SECTION

Educating the New Makers: Cross-Disciplinary Creativity

Mark D. Gross and Ellen Yi-Luen Do

ABSTRACT

Making computationally embedded things demands cross-disciplinary creativity, and creative designers must master many materials and methods. The studio-laboratory workspace is essential for learning to engage in such creative endeavors. In this kind of environment, students are encouraged to define their own problem statements and decide what to design. The faculty encourages tinkering, design and the play instinct. In this paper, the authors present their interest in building methods and tools that can open new design spaces in the studio-laboratory environment. They reflect on the distinctive characteristics of this learning environment and how these qualities aid design and foster creative engagement.

The concept of "thinking with your hands" and the use of computation as a new medium for making things can help us understand how to educate the "creative designer" in the age of ubiquitous and embedded computing. We reflect here on the characteristics of environments for learning creativity in the design of things with computationally mediated behavior. This paper describes basic characteristics of learning environments for design and patterns for creative engagement that can foster cross-disciplinary creativity in an academic setting.

THE "MAKER CULTURE"

The idea of the "maker culture" is not new. Why then this renewed interest in creativity and making? The coming decades will be an age in which consumers are becoming co-creators [1]. Examples are abundant. Teenagers are creating their own on-line content (e.g. Facebook) and the commercial success of Build-A-Bear Workshop retail outlets for "create-your-own" toy bears are testimonials to this phenomenon. Although these examples may seem trivial, they illustrate what seems to be a popular trend: More and more people want to become active participants and act as designers to contribute to personally meaningful activities. As Fischer noted in his article "Beyond Couch Potatoes" [2], we are once again in a "makers culture."

The discussion that follows is rooted in our own experience teaching in universities in North America, admittedly a limited context and certainly not the only place where creativity can or should be learned. Yet, in industrialized nations, a substantial fraction of people enjoy university (post-secondary) education, and this is therefore a point of leverage. Those who graduate from a university are likely to have opportunities to affect the larger social context. Our universities expect graduates to be leaders and innovators in the workplace and in society at large. By constructing their learning experience in certain ways, we hope to have a longer-term effect on how they see the world and, by extension, how they act in it.

In this paper we discuss some characteristics of the learning environments that we have created for educating designers who are intrepid boundary-crossers and able to function comfortably in a variety of domains: from aesthetic, emotional and artistic to functional and elegantly engineered. We encourage designers to transcend boundaries between fields and to explore by constructing interactions in material, software and hardware, in a process that creates, in Seymour Papert's phrase, "objects to think with" [3]. This approach is deeply embedded in design studio culture. We set up an environment to foster creative mindsets and approaches, specifically the process of generating ideas and building prototypes (see Article Frontispiece) to understand them.

We know that, for some, the positions we take here are quite familiar, perhaps even obvious. Yet, from where we sit, with some notable exceptions, these ideas do not appear to be widely adopted and embedded in university learning environments. It is in that context that we make our remarks.

Makers Know Materials and Processes

If creativity is rooted in making things, then to foster creativity we must look at how people learn to make things and to make them well. The curriculum of a design school (such as the Bauhaus foundation courses of the 20th century or, more recently, the programs of contemporary schools of architecture or industrial design) reveals an emphasis on materials and process. A potter must know clays and glazes and the various processes by which to prepare, form and fire them. A clothing designer must know fabrics and fasteners, sizing, cutting and sewing. Programmers too must know materials and processes: hardware and software and the procedure by which code is designed, written, debugged and maintained. Mastery of materials and processes—obtained through direct experience—is fundamental to making things in any domain.

The Studio-Laboratory Work Space Is Essential

Designers in all disciplines are accustomed to the studio model of learning and practice in which they play a role that Don Schön described in *The Design Studio* [4] as that of "reflective practitioner" [5]. Like the research laboratory in the sciences, the design studio depends on a common space where work takes place and is visible for informal discussion and open critique. A shared space for work is a basic ingredient of a creative community. However, this model is not found across the university; for example, the lab or studio is virtually unknown in the humanities [6].

Mark D. Gross (maker, educator, designer), Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A. E-mail: <mdgross@cmu.edu>.

Ellen Yi-Luen Do (maker, educator, designer), Georgia Institute of Technology, Atlanta, GA 30332-0155, U.S.A. E-mail: <ellendo@cc.gatech.edu>.

Based on a paper presented at the sixth Creativity and Cognition conference (13–15 June 2007, Washington, DC), on the theme "Materialities for Creativity," focused on the cultivating and sustaining of creativity.

Article Frontispiece. The authors' studio-laboratory has developed a spectrum of interactive prototypes and speculative designs. (© Ellen Yi-Luen Do)

Patterns of Creative Engagement

We describe three patterns that we have found useful in building computationally enhanced artifacts as objects to think and play with to explore and refine design concepts. They are: (1) owning the problem, (2) design and the play instinct and (3) building tools to make things. We illustrate each briefly with a student project.

Owning the Problem or Deciding What to Design: Gesture Modeling

An example of "owning the problem" is our Gesture Modeling project [7]. It began with the frustration of using screen-based interfaces to model in a CAD program. A designer wanted instead to use his hands to generate three-dimensional form. He wanted to design with computers as freely as he sculpted in clay. Gesture Modeling connected a 3D geometry engine with image processing code that recognized different hand gestures, thereby linking gestures to modeling operations. Figure 1 shows the project: A designer gestures to deform a mesh that represents a landscape. An overhead camera captures pictures of the hand, and software interprets the gestures; these in turn apply as operations to the mesh displayed on the screen.

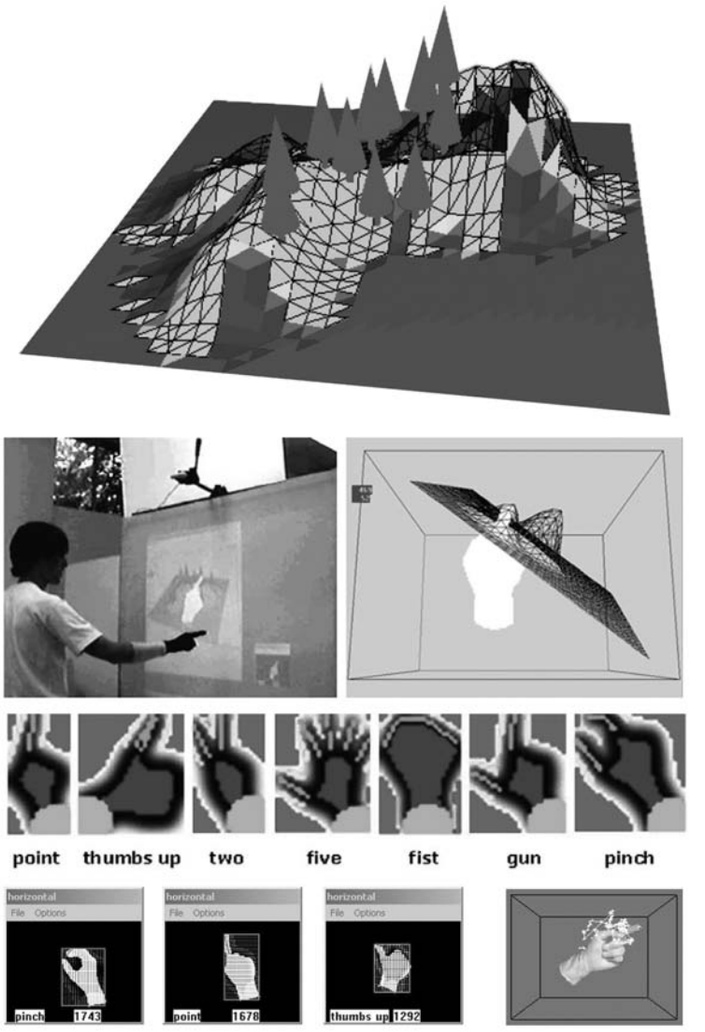


Fig. 1. Gesture modeling: deforming a mesh model with a hand gesture. (© Ellen Yi-Luen Do)

In interaction design it is customary to begin with a user-centered approach that identifies the dimensions of a problem to be solved. In a traditional architectural design studio, work begins with a clearly defined problem statement or "program" (e.g. a community library, a house for a working couple or a train station). This way of working is appropriate where the goal is to teach and learn specific skills that every architect must know, such as arranging functions in a floor plan or deciding on a structural system to support the building. The drawback—from the more general perspective of learning to make things—is that being given a ready-made problem avoids the framing question of "deciding what to design" [8]. The decisions of "what to design" exist at all levels of specificity. Owning the problems helps drive projects through both problem-seeking and problem-solving activities effectively.

As von Hippel shows in *Democratizing Innovation* [9], many innovations come about because the inventor solves a problem for him- or herself. Whereas professional designers must learn to conduct ethnographic studies and perform cultural probes in order to understand the users' context and needs, a designer who owns the problem has an inherent understanding of the problem and therefore an advantage in understanding what might be suitable to address it. In *Hackers and Painters,* Paul Graham puts it as follows:

> You're most likely to get good design if the intended users include the designer himself. When you design something for a group that doesn't include you, it tends to be for people you consider to be less sophisticated than you, not more sophisticated [10].

Therefore, we encourage designers to define their own problem statements—figuring out the "wants." We draw on personal experience and personal needs as a primary source for creative exploration of the design space. For example, one might be frustrated with existing technology or practice and want something better. Wants can also come from personal experience, from the desire to live a smarter, more efficient, or happier life. Having wants ensures a passion for something to happen. This motivates people to engage in just-in-time learning to achieve their project goals. The process begins with the egocentric ("I want . . . ") and moves toward a shared vision of the benefits of a project ("We get . . ."). Buchanan [11] points out that the old design education focuses on "teaching the materials, tools, and techniques of design as the primary subject matter," whereas the new "focuses on projects and problems that are situated within the experience and motivation of students." He argues, "Having a reason to design gives focus and purpose to student development. When a purpose exists, we find it easier then to introduce materials, tools, and techniques."

Tinkering, Design and the Play Instinct: Bach Blocks

A second pattern is play. The Bach Blocks (Fig. 2) comprise a set of colored blocks, a camera and software that reads the

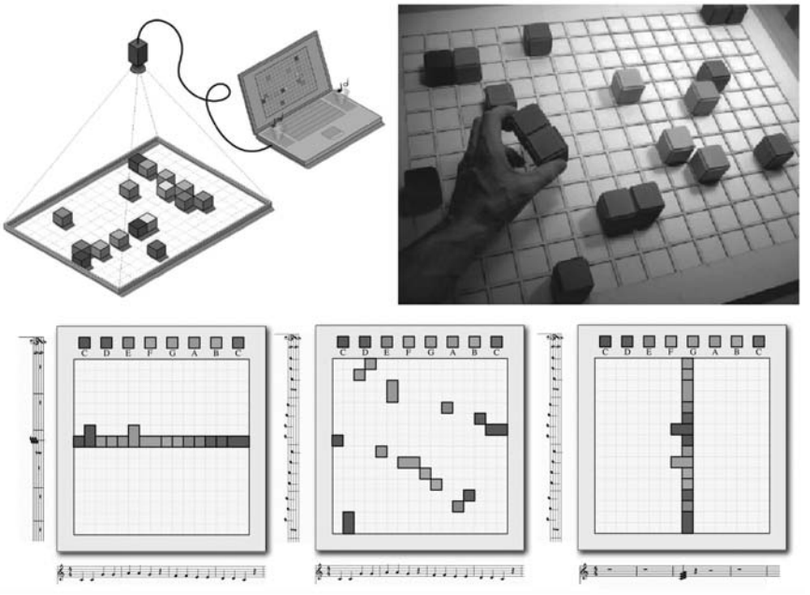


Fig. 2. Bach Blocks: making and playing music with colored blocks. (© Ellen Yi-Luen Do)

arrangement of blocks to play a tune. The designer wanted to play with music and to build a toy to engage his young children with music. Colors represent pitches and the positions of the blocks determine the sequence of play. Thus, Bach Blocks is at once an instrument for making music and a notation to compose it. Ordinarily the software plays from left to right (and blocks arranged vertically play harmonies), but Bach Blocks can also be set to play the tune in any direction.

The late American graphic designer Paul Rand described designing as a kind of play within given or self-imposed constraints [12]. Play—an exploration of materials and processes—is what distinguishes routine acts of production from more creative acts of making that may result in innovative ideas. Papert's *bricolage,* or "hacking," goes to the heart of what creative people often do and what people who aspire to being creative must practice. Schools of design and the arts encourage this sort of creative play, those of engineering and computer science not as much.

In engineering and computer science education, hacking, tinkering and playful exploration are often disparaged. They are seen as insufficiently goal-oriented: A good engineer, it is said, begins with a clearly articulated problem statement and then applies reliable methods to reach a solution. Playing around with things wastes time that could be more profitably spent applying known methods to the problem at hand, and, inevitably, there will be failures, as actually building a prototype reveals unanticipated behaviors that undermine a previously plausible design idea. Getting the plan right in the first place avoids wasted time and costly mistakes.

Play is important because creative design seldom begins with a clearly stated problem. Rather, as many have pointed out [13,14], design is as much concerned with identifying and expressing a problem as it is with finding solutions. Well-worn adages such as "Defining the problem *is* the problem" and "The problem and solution co-evolve" exemplify this characteristic of designing.

Building Tools to Make Things: Furniture Factory

A third pattern is tool-building. We are less interested in making a design for a particular client or user than in developing ways of working—methods and tools—that can open new design spaces. The Furniture Factory program [15] helps designers make physical prototypes using rapid prototyping and manufacturing machines. A designer can use its sketch-based interface to draw furniture in a 3D isometric view. The program then displays the model in an Open-GL viewing window and decomposes the 3D model into flat panels and displays these parts (Fig. 3). Furniture Factory adds joints according to the connection conditions where panels meet. These added joints enable designers to construct the physical model easily and quickly. The program then generates code to produce the furniture parts on a laser cutter. The Furniture Factory program is a tool for designers to sketch and manufacture a simple subset of the universe of things that can be made from flat material.

Educating the New Makers

Everyone can be creative, because everyone can make things. Creativity is rooted in the experience of making things; mastery of materials and processes is fundamental. Self-motivation, playfulness and a willingness to transcend boundaries are also key qualities. They are much needed in educating a class of designers that we might call "the New Makers." These designers are at home in designing the artifacts and systems that may have both physical and material characteristics as well as computational functions and behavior. Yet formal education in engineering and computer science works against these qualities. Traditional engineering curricula focus on teaching analysis; design typically appears at the end of study, in a "capstone" course. Although solving design problems in capstone engineering courses does offer opportunities to be creative, they are tightly framed with a limited range for exploration. Traditional design studio courses in architecture and industrial design allow more room for exploration, but still the goal is to solve a problem the instructor sets. Similarly, problem-based learning tightly structures the design space, in the interest of maintaining focus and achieving results in a prescribed time and space.

Our position is simple. To foster creativity we must teach people to make things. People learn to make things through practice, experiment and playful prototyping. They work with specific materials and processes in a physical and social environment that encourages cross-talk and some critique among makers. Problem-seeking and problem-solving go hand in hand, and owning a problem is the strongest motivator. Finally, those who learn to make very different kinds of things gain powerful insights into creative processes.

An interdisciplinary, no-boundaries, technically enabled studio-laboratory is still unusual in the university. Happily we are not alone: Pelle Ehn articulates a similar vision for a Digital Bauhaus at Malmö University [16]; Stanford's "d-school," the ID-StudioLab at Delft, the MIT Media Lab and others are ventures in the same vein. Within the design disciplines, and especially industrial and interaction design, there is a growing interest in hybrid models of education [17–19]. At the grassroots level, the suc-

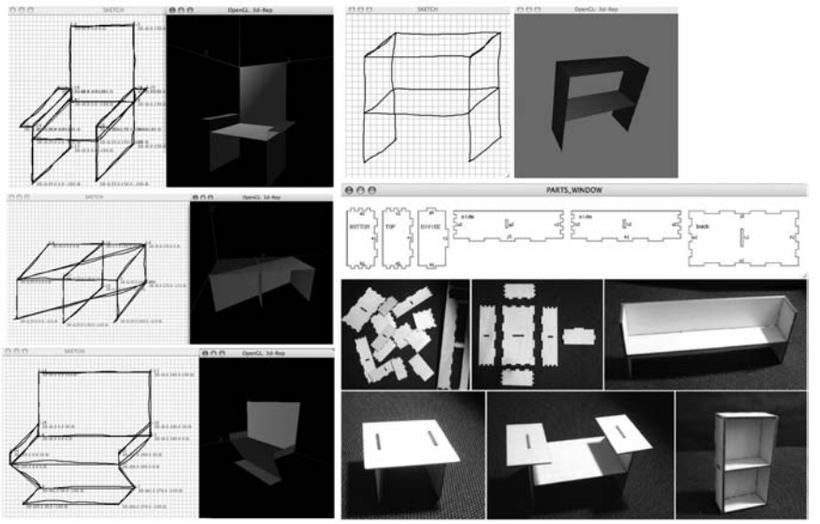


Fig. 3. Sketch to Fabrication with the Furniture Factory. (© Ellen Yi-Luen Do)

cess of *Make* magazine and its MakerFaire (which attracted tens of thousands of visitors to its San Francisco event in 2007) demonstrates widespread enthusiasm for this way of working. Still, the model runs against the grain of the university, which rewards focus *within* rather than *across* disciplines.

Education of Designers vs. That of Engineers

Designers are educated differently from scientists and engineers. Architects in particular are integrators and hence negotiators among a diverse range of other experts. Designers are taught to keep options open, explore parallel alternatives and embrace ambiguity. Engineers tend to be more goal-oriented, to stay within their field of expertise and see ambiguity as something to be eliminated. Thus, we find that it is easier for design students to acquire the technical skills (programming, electronics) they need to carry out projects with a computational component than for engineering and computer science students to acquire the mindset needed to work in ill-defined situations.

Engineering and computer science students tend to be less prepared for open-ended investigation than those who have studied and practiced design. Engineering and computer science students with whom we have worked are most comfortable when given a specification of work to be accomplished. An open-ended brief makes them uneasy—they do not know where to begin or how to proceed. Once a clear objective is stated they can apply their skills to attain it. Design students, by contrast, will ignore even a clear specification and do something else!

Perhaps gaining technical expertise is simply easier than learning to design. Perhaps other factors, however, are responsible for this phenomenon. One is cultural asymmetry: Due to a perceived hierarchy in the university, computer science and engineering students fail to recognize an important skill that they lack. Indeed, their education is designed to prepare them to walk up to any new problem and apply their bag of tricks to it.

In our experience a computer science or engineering student is more likely to jump from an initial problem statement to propose a method of solving it, then immediately pursue that approach without considering alternatives. In contrast, generating and comparing alternatives at every stage is drummed into designers throughout their education.

There is also the matter of simply being able to accomplish *something*. Usually a computer scientist or engineer who knows how to code can make something—however poorly designed or inelegant—without having first learned to design. On the other hand, a designer who sets out to make an artifact that employs software or hardware must—perforce—learn some technical skills.

Of course, not every designer finds programming a natural medium. Designers are sometimes overwhelmed by the technical detail that they must master in order to do anything interesting. Many who take computer science courses are, on the one hand, bored by the examples used in problem sets (which have no obvious relationship to anything they might be interested in) and, on the other hand, discouraged by the attention to detail that is needed to make a working piece of software—although they may be comfortable with this level of detail and craft in making drawings or physical objects. Other students view programming as a way to get something done—by any means necessary—and, failing to recognize the power of good design in software, produce horrible kludges that work (perhaps) for a key example or two but in the end limit exploration.

Programming as Designing (de.sign = pro.gram)

If creativity is about making things, and making things is about design, what is the place of programming in all this? Many who have worked both as a programmer and in some other domain of design recognize powerful parallels.

Consider the words of master programmer Dick Gabriel. In 2004 Gabriel received the AAAI/ACM Allen Newell Award "for innovations not only on fundamental issues in programming languages and software design but also on the interaction between computer science and other disciplines, notably architecture and poetry." In an interview titled "The Poetry of Programming," Gabriel said:

> Writing code certainly feels very similar to writing poetry. When I'm writing poetry, it feels like the center of my thinking is in a particular place, and when I'm writing code the center of my thinking feels in the same kind of place. It's the same kind of concentration. So, I'm thinking up possibilities, I'm thinking about, well, so how do I reinvent the code, gee, you know, what's the simplest way to do this [20].

Or, to quote Paul Graham again:

> Hacking and painting have a lot in common. In fact, of all the different types of people I've known, hackers and painters are among the most alike. What hackers and painters have in common is that they're both makers. Along with composers, architects, and writers, what hackers and painters are trying to do is make good things [21].

Martin Brynskov at the Center for Interactive Spaces at the University of Aarhus [22] points out that the words *design* and *program* are remarkably close in their Greek and Latin roots. According to the *Oxford English Dictionary,* the word *design* is made from the prefix *de-* (meaning out) and the root *sign* (meaning mark) that is, to mark out. Likewise, the word *program* is made from the prefix *pro-* (meaning forward or out), and *gram* (meaning writing). That is, both design and program mean to mark out or make an explicit representation.

Emphasis on Prototypes

We emphasize building working prototypes of ideas quickly. These abstract out and focus on parts of an idea that seem interesting or worth exploring. In traditional design, prototypes are physical models that illustrate the form and perhaps the materials of the design to be made. As computational behavior becomes more important, it becomes difficult for traditionally trained designers to make working prototypes that illustrate not only the physical and material characteristics of designs but also their functional behavior. As designers grapple with making things that have both physical and computational characteristics, the need becomes apparent for prototyping tools that can capture and convey not only the physical manifestation of a design, but also functional and behavioral ones.

Toolkits and design environments for physically embedded computation [23–25] can make this kind of work more accessible.

The New Makers

Today, many argue that creativity is crucial in the new economy [26]. Perhaps we can foster creativity in society by putting making back into education. There is nothing new about this idea, although learning to make things has become conspicuously absent in most courses of higher education. One might expect schools of engineering to teach students to make things, but engineering curricula are heavy on analysis and principles and light on making. Making things is still taught in two places in the university: schools of design and the arts, and departments of computer science. Surprisingly, as computer science matures as a discipline, the skills of making software are being displaced by an emphasis on more analytic skills. Even in "capstone design" courses one seldom finds open-ended design problems. Perhaps it is easier to assess performance on closed-form problems, or perhaps, simply because there is now more to know, computer science is moving away from design, just as other engineering disciplines such as mechanical or civil engineering did in their earlier days.

Now is an interesting moment. Things have changed, and the ways of making things have changed too. Almost everything in our world is designed. Increasingly, designing is mediated by computational processes and the artifacts that we encounter—our shoes, our houses, even our parks—are embedded with microcontrollers, sensors and electronics. Designers of the future—the New Makers—will need to be fluent with the materials and processes of computation, in addition to the materials and processes of other domains.

Powerful insights are available to those designers—in any discipline—who master the art and craft of making things in more than one domain. These insights may eventually further what Simon termed a "Science of Design." Meanwhile, we hold that creativity is rooted in, and therefore best learned through, the experience of making things.

Acknowledgments

An extended version of this paper appeared in the *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition 2007.* This material is based upon work supported by the National Science Foundation under Grant ITR-0326054 and the Pennsylvania Infrastructure Technology Alliance.

References and Notes

Unedited references as provided by the authors.

1. E.B.-N. Sanders, Information, Inspiration and Cocreation. 2005 [accessed August 3, 2008]; <http://maketools.com/pdfs/InformationInspirationandCocreation_Sanders_05.pdf>.

2. G. Fischer, Beyond Couch Potatoes: From Consumer to Design, in Proceedings, 3rd Asia Pacific Computer Human Interaction (APCHI). 1998. pp. 2–9.

3. S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas.* 1980, New York: Basic Books.

4. D. Schön, *The Design Studio.* 1985, London: RIBA.

5. D.A. Schön, "Learning a Language, Learning to Design," in Architecture Education Study, W.L. Porter and M. Kilbridge, Editors. 1981, Andrew W. Mellon Foundation: New York. pp. 339–471.

6. G. Hiatt, "We Need Humanities Labs." 2005 October 26 [accessed August 3, 2008]; <www.insidehighered.com/views/2005/10/26/hiatt>.

7. A. Kemp and M.D. Gross. "Gesture Modelling: Using Video to Capture Freehand Modeling Commands," in Computer Aided Architectural Design Futures. 2001. Eindhoven, NL: Kluwer. pp. 33–46.

8. M. Shaw, J. Herbsleb, and I. Ozkaya. "Deciding What to Design: Closing a Gap in Software Engineering Education," in Proceedings of the 27th International Conference on Software Engineering (ICSE). 2005: ACM. p. 607–608.

9. E. von Hippel, Democratizing Innovation. 2005, Cambridge, MA: MIT Press.

10. P. Graham, Hackers and Painters. 2004, Sebastopol, CA: O'Reilly.

11. R. Buchanan, Design Research and the New Learning. Design Issues, 1999. 17(4): pp. 3–23.

12. P. Rand, "Design and the Play Instinct," in Education of Vision, G. Kepes, Editor. 1965, Braziller: New York. pp. 154–173.

13. H. Ritteland and M.M. Webber, "Dilemmas in a General Theory of Planning." Policy Sciences, 1969. 4(1973): pp. 155–169.

14. H. Simon, *Sciences of the Artificial.* 1969, Cambridge, MA: MIT Press.

15. Y. Oh, G. Johnson, M. D. Gross, E.-Y. Do. "The Designosaur and the Furniture Factory: Simple Software for Fast Fabrication," in Proceedings Design Computing and Cognition '06. 2006: Kluwer. pp. 123–140.

16. P. Ehn, *A Manifesto for a Digital Bauhaus.* Digital Creativity, 1998. 9(4): pp. 207–217.

17. S.R. Klemmer, B. Verplank, and W. Ju. "Teaching Embodied Interaction Design Practice," in Proceedings of the 2005 conference on Designing for User eXperience. 2005. San Francisco, California: AIGA: American Institute of Graphic Arts.

18. S. Lundgren, et al. Teaching Interaction Design: Matters, Materials and Means. 2006 [accessed 20 July 2008]; <www.cs.chalmers.se/idc/publication/pdf/lundgren_et_al_wonderground.pdf>.

19. K. Sato and W. Verplank, Panel: Teaching Tangible Interaction Design, in Designing Interactive Systems: processes, practices, methods, and techniques. 2000, ACM Press: New York City, New York, United States. pp. 444–445.

20. R. Gabriel, The Poetry of Programming. 2002 [accessed Oct 16, 2006]; <http://java.sun.com/features/2002/11/gabriel_qa.html>.

21. Graham [10] p. 18.

22. Personal Communication, M. Brynskov, Pittsburgh. February, 2006

23. B. Hartmann, et al., "Reflective physical prototyping through integrated design, test, and analysis," in Proceedings of the 19th annual ACM Symposium on User Interface Software and Technology. 2006, ACM Press: Montreux, Switzerland. pp. 299–308.

24. S.E. Hudson and J. Mankoff, "Rapid construction of functioning physical interfaces from cardboard, thumbtacks, tin foil and masking tape," in Proceedings of the 19th annual ACM Symposium on User Interface Software and Technology. 2006, ACM Press: Montreux, Switzerland. pp. 289–298.

25. J.C. Lee et al. "The calder toolkit: wired and wireless components for rapidly prototyping interactive devices," in Designing interactive systems: processes, practices, methods, and techniques. 2004: ACM. pp. 167–175.

26. R. Florida, The Rise of the Creative Class: And How It's Transforming Work, Leisure, Community and Everyday Life. 2003, New York: Basic Books.

Manuscript received 2 February 2008.

Mark D. Gross is Professor of Computational Design at Carnegie Mellon University, equally at home writing software and practicing architectural design-build on his own houses.

Trained as an architect in Taiwan, Ellen Yi-Luen Do began working on projects combining design and computation after her arrival in the United States in 1990. She is currently an associate professor in the College of Architecture and the College of Computing at Georgia Institute of Technology, working toward making intuitive design tools, things that think, spaces that sense and places that play.