Epistemological Pluralism: Styles and Voices within the Computer Culture
Author(s): Sherry Turkle and Seymour Papert
Source: *Signs*, Vol. 16, No. 1, From Hard Drive to Software: Gender, Computers, and Difference (Autumn, 1990), pp. 128-157
Published by: The University of Chicago Press
Stable URL: http://www.jstor.org/stable/3174610
Accessed: 05-12-2017 22:56 UTC

EPISTEMOLOGICAL PLURALISM: STYLES AND VOICES WITHIN THE COMPUTER CULTURE

SHERRY TURKLE AND SEYMOUR PAPERT

Epistemological pluralism

The prevailing image of the computer represents it as a logical machine and computer programming as a technical, mathematical activity. Both the popular and technical culture have constructed computation as the ultimate embodiment of the abstract and formal. Yet the computer's intellectual personality has another side: our research finds diversity in the practice of computing that is denied by its social construction. When we looked closely at programmers in action we saw formal and abstract approaches; but we also saw highly successful programmers in relationships with their material that are more reminiscent of a painter than a logician. They use concrete and personal approaches to knowledge that are far from the cultural stereotypes of formal mathematics.[1]

[1] Research reports that emphasize approach to programming or programming style in the sense we are using it here include Seymour Papert, Andrea di Sessa, Sylvia Weir, and Daniel Watt, "Final Report of the Brookline Logo Project," Logo Memos 53 and 54 (Massachusetts Institute of Technology, Cambridge, Mass., 1979); Sherry Turkle, "Computer as Rorschach," *Society* 17 (December 1980): 15–22, and *The Second Self: Computers and the Human Spirit* (New York: Simon & Schuster, 1984), esp. chap. 3; Sylvia Weir, *Cultivating Minds: A Logo Casebook* (New York: Harper & Row, 1987); Sherry Turkle, Donald Schon, Brenda Nielsen, M. Stella Orsini, and Wim Overmeer, "Project Athena at MIT" (Massachusetts Institute of Technology, Cambridge, Mass., May 1988, typescript); Lise Motherwell, "Gender

[*Signs: Journal of Women in Culture and Society* 1990, vol. 16, no. 1]

The diversity of approaches to programming suggests that equal access to even the most basic elements of computation requires accepting the validity of multiple ways of knowing and thinking, an epistemological pluralism. Here we use the word epistemology in a sense closer to Piaget's than to the philosopher's.[2] In the traditional usage, the goal of epistemology is to inquire into the nature of knowledge and the conditions of its validity; and only one form of knowledge, the propositional, is taken to be valid.[3] The step taken by Piaget in his definition of *epistemologie genetique* was to eschew inquiry into the "true" nature of knowledge in favor of a comparative study of the diverse nature of different kinds of knowledge, in his case the kinds encountered in children of different ages. We differ from Piaget on an important point, however. Where he saw diverse forms of knowledge in terms of stages to a finite end point of formal reason, we see different approaches to knowledge as styles, each equally valid on its own terms.

The barriers to acknowledging such pluralism are great, historically rooted in domains that go far beyond computation. The formal, propositional way of knowing has been recognized traditionally as a standard, canonical style. Indeed, philosophical epistemology has generally taken it as synonymous with knowledge. Where concrete approaches to knowledge have been recognized at all, it has most often been as inferior ways of knowing, the kinds of knowing adopted by necessity by those who have not yet mastered the canonical style. Thus Jean Piaget recognizes in young children ways of thinking that do not conform to the canon but that are too coherent and efficacious to be branded simply as "wrong." He casts children's concrete thinking as a stage in a progression to a formal style.[4] Similarly, Claude Lévi-Strauss recognizes "bricolage," a

---

and Style Differences in a Logo-based Environment" (Ph.D. diss., Massachusetts Institute of Technology, January 1988); Idit Harel, "Software Design for Learning: Children's Construction of Meaning for Fractions and Logo Programming" (Ph.D. diss., Massachusetts Institute of Technology, June 1988).

[2] Piaget's use of the term epistemology is pervasive in his writing. See, in particular, *Introduction à epistemologie genetique*, vols. 1–3 (Paris: Presses Universitaires de France, 1950). Alvin Goldman discusses the modern redefinition of the field of epistemology as something as close to psychology and sociology as to philosophy in *Epistemology and Cognition* (Cambridge, Mass.: Harvard University Press, 1986).

[3] For a critical and polemical account of this history, see Paul M. Churchland, *A Neurocomputational Perspective: The Nature of Mind and the Structure of Science* (Cambridge, Mass.: MIT Press, 1989).

[4] See, e.g., Jean Piaget and Barbel Inhelder, *The Growth of Logical Thinking from Childhood to Adolescence* (New York: Basic, 1958).

"science of the concrete," but relegates it to primitive societies, a manifestation of the "savage mind."[5]

More recently, concrete ways of thinking have been recognized in contexts that are not easily dismissed as inferior. Ethnographers of science studying the daily life of the laboratory have found that scientific discoveries are made in a concrete, ad hoc fashion, and only later recast into canonically acceptable formalisms.[6] Scientific biography reveals that Nobel laureates relate to their materials in the concrete and tactile style of Lévi-Strauss's bricoleurs.[7] Psychologists investigating adults' mathematical thinking find that they use an effective and down-to-earth style very different from the abstract and formal math they were taught at school.[8] Feminist scholars have documented the power of concrete, contextual reasoning in a wide range of domains.[9]

[5] Claude Lévi-Strauss, *The Savage Mind* (Chicago: University of Chicago Press, 1968).

[6] A sample of relevant studies in scientific ethnography is provided by Karin Knorr-Cetina and Michael Mulkay, eds., *Science Observed: Perspectives on the Social Studies of Science* (London: Sage, 1983). See also Karin Knorr-Cetina, *The Manufacture of Knowledge: An Essay on the Constructivist and Contextual Nature of Science* (Oxford: Pergamon, 1981); Bruno Latour and Stephen Woolgar, *Laboratory Life: The Social Construction of Scientific Facts* (Beverly Hills, Calif.: Sage, 1979); Sharon Traweek, *Beamtimes and Lifetimes: The World of High Energy Physicists* (Cambridge, Mass.: Harvard University Press, 1989).

[7] Evelyn Fox Keller, *A Feeling for the Organism: The Life and Work of Barbara McClintock* (San Francisco: W. H. Freeman, 1983).

[8] A sample of studies on everyday thinking is contained in Barbara Rogoff and Jean Lave, eds., *Everyday Cognition: Its Development in Social Context* (Cambridge, Mass.: Harvard University Press, 1984). Also, see Jean Lave, *Cognition in Practice: Mind, Mathematics and Culture in Everyday Life* (Cambridge: Cambridge University Press, 1988).

[9] See e.g., Mary Field Belenky, Blythe McVicker Clinchy, Nancy Rule Goldberger, and Jill Mattuck Tarule, *Women's Ways of Knowing: The Development of Self, Voice, and Mind* (New York: Basic, 1986). Edited collections that focus on approaches to knowing in science include: Ruth Bleir, ed., *Feminist Approaches to Science* (New York: Pergamon, 1986); and Sandra Harding and Merrill B. Hintikka, eds., *Discovering Reality: Feminist Perspectives on Epistemology, Metaphysics, Methodology, and Philosophy of Science* (London: Reidel, 1983). An overview that highlights many of the issues we deal with in this essay is provided by Elizabeth Fee, "Critiques of Modern Science: The Relationship of Feminism to Other Radical Epistemologies," in Bleir, ed. In this essay we situate our position by focusing on two writers, Carol Gilligan and Evelyn Fox Keller. Gilligan, with her emphasis on moral discourse, might seem out of place in a discussion of noncanonical approaches to science and technology; but here we argue that key issues in the critique of science are not about *scientific* reasoning but about *reasoning*. Juxtaposing moral and computational reasoning helps us make this point. In addition, Gilligan's critical relationship to the theories of Lawrence Kohlberg is analogous to our own critical relationship to Piaget's work. We emphasize Keller because her work underscores, as does ours, the importance of relationships with objects in the development of noncanonical styles. Using Gilligan and Keller as a contrasting pair allows us to

With such contributions has come a growing convergence of intellectual commitments to a revaluation of the concrete; but in general, the ethnographers, psychologists, and feminist scholars who have contributed to this revaluation have not seen computation as relevant to their concerns. Here we present evidence that points toward the possibility of new intellectual alliances.

In our research on programming styles, the computer has emerged as an important actor in the revaluation of the concrete, a privileged medium for the growth of alternative voices in dealing with the world of formal systems. The conventional route into formal systems, through the manipulation of abstract symbols, closes doors that the computer can open. The computer, with its graphics, its sounds, its text, and its animation, can provide a port of entry for people whose chief ways of relating to the world are through movement, intuition, and visual impression. At the heart of the new possibilities for the appropriation of formal systems is the computational object, on the border between an abstract idea and a concrete physical object. In the simplest case, a computational object such as an icon moving on a computer screen can be defined by the most formal of rules and is thus a mathematical construct, but at the same time it is visible, almost tangible, and allows a sense of direct manipulation that only the encultured mathematician can feel in traditional formal systems.[10] The computer has a theoretical vocation: it can make the abstract concrete; it can bring formality down-to-earth.

We have studied computers and the cultures that grow up around them in a wide variety of settings ranging from video game arcades to research laboratories of artificial intelligence. In this paper we draw particularly on a long-term line of research on how people enter the culture of programming. Using clinical methods inspired by the Piagetian and psychoanalytic traditions, we built up case studies of children using computers in grade school settings and college students taking a first programming course. We saw many manifestations of the concrete approach, favored in our study by more women than men. We were also able to observe people reacting poignantly to what they felt as a pressure to conform to an

highlight two different dimensions of what we call the concrete approach to science (see Carol Gilligan, *In A Different Voice: Psychological Theory and Women's Development* [Cambridge, Mass.: Harvard University Press, 1982]; Evelyn Fox Keller, *A Feeling for the Organism,* and *Reflections on Gender and Science* [New Haven, Conn.: Yale University Press, 1985]).

[10] See Philip J. Davis and Reuben Hersh, *The Mathematical Experience* (Boston: Houghton Mifflin, 1981); and Seymour Papert, "The Mathematical Unconscious," in *Aesthetics and Science,* ed. Judith Wechsler (Cambridge, Mass.: MIT Press, 1980).

officially imposed style.[11] Although the computer as an expressive medium supports epistemological pluralism, the computer culture often does not. Our data points to discrimination in the computer culture that is determined not by rules that keep people out but by ways of thinking that make them reluctant to join in. Moreover, the existence of diverse styles of *expert* programming supports the idea that there can be different but equal voices even where the formal has traditionally appeared as almost definitionally supreme: in mathematics and the sciences.

Evelyn Fox Keller has remarked on the difficulty that people face when they try to understand what it might mean to do science in anything other than the formal and abstract canonical style. Describing such a style in the work of geneticist Barbara McClintock, Keller notes that this is the "less accessible aspect" of a scientist's relationship to nature.[12] In this essay we describe people learning to program who are having experiences with formal systems that are in many ways analogous to those of the bricoleur scientist or mathematician. One way the computer contributes to the revaluation of concrete approaches in the domain of formal systems is by giving more people access to (and an experience of) them.

The computer forces general questions about intellectual style to reveal an everyday face.[13] Even schoolroom differences in how children program computers raise issues that come up in a more abstract form in scholarly debates about scientific objectivity. The computer makes ideas about noncanonical scientific voices more concrete and therefore appropriable because we can relate them not only to the science of the scientists but to our own thinking.

Here we focus on descriptions of a concrete way of knowing; the formal, canonical style is well known and well defended. Yet, our discussion of concrete approaches is implicitly a discussion of

[11] For grade school children we worked with forty cases. Of the twenty girls in our study, fourteen preferred concrete approaches, of twenty boys there were four who followed this route. In the study of college students taking a first programming course, of the fifteen women, nine were concrete style programmers, of fifteen men, four. Because of our interest in spontaneous approaches, we classified as concrete thinkers some students who finally adopted elements of the canonical approach in order to please their teachers. (See, e.g., the cases of Lisa and Robin, below.) In our research, the male/formal and female/concrete dichotomy was most dramatic in a predominantly white, wealthy private school in the South where traditional patterns of socialization would favor boys learning the ways of control, hierarchy, and distance and girls learning the ways of negotiation and closeness.

[12] Keller, *A Feeling for the Organism,* 198.

[13] For a fuller discussion of the computer as an evocative and concretizing object, see Turkle, *The Second Self* (n. 1 above).

formal ones; it contributes to the deconstruction of the canonical style as the only way to think. It also situates it: the supervaluation of the formal approach owes much of its strength within computation to the support it gets in other intellectual domains. Formal thinking, defined as synonymous with logical thinking, has been given a privileged status that can be challenged only by developing a respectful understanding of other styles, where logic is seen as a powerful instrument of thought but not as the "law of thought." In this view, "logic is on tap not on top." As a carrier for pluralistic ideas about approaches to knowledge, the computer may hold the promise of catalyzing change not only within the computer culture but in the culture at large.

Personal appropriation

Consider Lisa, eighteen, a first-year Harvard University student in an introductory programming course. Lisa fears that she will find the course difficult because she is a poet, "good with words not numbers." In high school, she had always scorned teachers who had insisted that mathematics is a language. Yet, now, her first encounter with the computer has made Lisa ready to reconsider this proposition and with it her characterization of herself as someone "bad at math." Lisa starts well, surprised to find herself easily in command of the course material; but as the term progresses she reluctantly decides that she "has to be a different kind of person with the machine." The pressure to do so is not from the computational medium. She says she can no longer resist pressure from her teachers to think in ways that are not her own.

Lisa wants to manipulate computer language the way she works with words as she writes a poem. There, she says, she "feels her way from one word to another," sculpting the whole. When she writes poetry, Lisa experiences language as transparent, she knows where all the elements are at every point in the development of her ideas. She wants her relationship to computer language to be similarly transparent. When she builds large programs she prefers to write her own, smaller, building block procedures even though she could use prepackaged ones from a program library; she resents the opacity of prepackaged programs. Her teachers chide her, insisting that her demand for transparency is making her work more difficult; Lisa perseveres, insisting that this is what it takes for her to feel comfortable with computers.

Two months into the programming course, Lisa's efforts to succeed are no longer directed toward trying to feel comfortable. She has been told that the "right way" to do things is to control a

program through planning and black-boxing, the technique that lets you exploit opacity to plan something large without knowing in advance how the details will be managed. Lisa recognizes the value of these techniques—for someone else. She struggles against using them as the starting points for her learning. Lisa ends up abandoning the fight, doing things "their way," and accepting the inevitable alienation from her work. She calls her efforts to become "another kind of person with the machine" her "not-me strategy" and begins to insist that the computer is "just a tool." "It's nothing much," she says, "just a tool." Lisa's growing sense of alienation does not stem from an inability to cope with programming but from her ability to handle it in a way that comes into conflict with the computer culture she has entered.

A classmate, Robin, is a pianist. Robin explains that she masters her music by perfecting the smallest "little bits of pieces" and then building up. She cannot progress until she understands the details of each small part. Robin is happiest when she uses this tried and true method with the computer, playing with small computational elements as though they were notes or musical phrases. Like Lisa, she is frustrated with using prepackaged programs. She, too, has been told her way is wrong: "I told my teaching fellow I wanted to take it all apart and he laughed at me. He said it was a waste of time, that you should just black box, that you shouldn't confuse yourself with what was going on at that low level."

Lisa and Robin came to the programming course with anxieties about not belonging (fearing that the computer belonged to male hackers who lived in "a world apart"), and their experiences in it only served to validate their fears.[14] Although carefully designed and imaginative, the Harvard University course taught that there is only one right way to approach the computer, a way that emphasizes control through structure and planning. There are many virtues to this computational approach (it makes sense when dividing the labor on a large programming project, for instance) but Lisa and Robin have intellectual styles at odds with it. Lisa says she has "turned herself into a different kind of person" in order to

[14] Lisa and Robin were part of a larger study of Harvard and MIT students taking introductory programming courses. The study found anxiety about an identity as a "computer person" to be an important aspect of reticence toward computers, especially among women (see Sherry Turkle, "Computational Reticence: Why Women Fear the Intimate Machine," in *Technology and Women's Voices: Keeping in Touch*, ed. Cheris Kramarae [New York: Pergamon, 1988]). See also Sara Kiesler, Lee Sproull, and Jacquelynne S. Eccles, "Poolhalls, Chips, and War Games: Women in the Culture of Computing," *Psychology of Women Quarterly* 9, no. 4 (December 1985): 451–62.

perform, and Robin says she has learned to "fake it." Although both women got good grades in this programming course, both have had to deny who they are in order to succeed.

Lisa's and Robin's experiences make it clear that the computer can be a partner in a great diversity of relationships, that the computer is an expressive medium that different people can make their own in their own way. Yet those who wish to approach the computer in a noncanonical way are discouraged by the dominant computer culture, eloquently expressed in the ideology of the Harvard University course. They are asked to change their style to suit the fashion when they begin to interact with the official computer world, committed to a formal, rule-driven, hierarchical approach to programming.[15] Like Lisa and Robin, their exclusion from the computer culture is perpetuated not by rules that keep them out, but by ways of thinking that make them reluctant to join in. They are not computer phobic. They do not need to stay away because of fear or panic; but they are computer reticent. They want to stay away because the computer has come to symbolize an alien way of thinking. They learn to get by and to keep a certain distance. One of its manifestations is the way they neutralize the computer through language, which denies the possibility of using it creatively (recall how Lisa dismisses it as "just a tool").

In this way, discrimination in the computer culture takes the form of discrimination against approaches to knowledge, most strikingly against the one preferred by Lisa and Robin, an approach we call "bricolage."

Bricolage

Lévi-Strauss used the term "bricolage" to contrast the analytic methodology of Western science with what he called a "science of the concrete" in primitive societies.[16] The bricoleurs he describes

[15] In 1987, Turkle led a discussion group attended by thirty-seven women members of a local computer society. Of these, seventeen reported feeling pressure to change their preferred ways of working with the computer in order to be more acceptable to the dominant computer culture. "I got my wrist slapped enough times and I changed my ways," said one of them, a college student for whom programming on her Macintosh was a private passion until she entered MIT.

[16] Lévi-Strauss (n. 5 above). Lévi-Strauss contrasted bricolage with Western science, ignoring the significant aspects of bricolage present in the latter. Several recent writers have written in a way that begins to redress this imbalance (see, e.g., Paul Feyerabend, *Against Method: The Outline of an Anarchistic Theory of Knowledge* [London: New Left Books, 1975]; N. R. Hanson, *Patterns of Discovery* [Cambridge: Cambridge University Press, 1958]; Ludwig Wittgenstein, *Philosophical Investigations* [New York: MacMillan, 1953]). In a less formal vein, see Richard Feynman, *Surely You Must Be Joking Mr. Feynman* (New York: Norton, 1985).

do not move abstractly and hierarchically from axiom to theorem to corollary. Bricoleurs construct theories by arranging and rearranging, by negotiating and renegotiating with a set of well-known materials.

Lévi-Strauss's descriptions of the two scientific approaches, divested of his efforts to localize them culturally, suggest the variety of ways that people approach computers. For some people in our study, what is exciting about computers is working within a rule-driven system that can be mastered in a top-down, divide-and-conquer way. This is the "planner's" approach taught in the Harvard programming course. This approach decrees that the right way to solve a programming problem is to dissect it into separate parts and design a set of modular solutions that will fit the parts into an intended whole. Some programmers work this way because their teachers or employers insist that they do. For others, it is a preferred approach; to them, it seems natural to make a plan, divide the task, use modules and subprocedures.

Lisa, Robin, and others like them in our study offer examples of a very different style. They are not drawn to structured programming; their work at the computer is marked by a desire to play with the elements of the program, to move them around almost as though they were material elements—the words in a sentence, the notes in a musical composition, the elements of a collage.

The bricoleur resembles the painter who stands back between brushstrokes, looks at the canvas, and only after this contemplation, decides what to do next. For planners, mistakes are missteps; for bricoleurs they are the essence of a navigation by mid-course corrections. For planners, a program is an instrument for premeditated control; bricoleurs have goals, but set out to realize them in the spirit of a collaborative venture with the machine. For planners, getting a program to work is like "saying one's piece"; for bricoleurs it is more like a conversation than a monologue. In cooking, this would be the style of those who do not follow recipes and instead make a series of decisions according to taste. While hierarchy and abstraction are valued by the structured programmers' planner's aesthetic, bricoleur programmers prefer negotiation and rearrangement of their materials.

For instance, Alex, nine years old, is a classic bricoleur. He attends the Hennigan Elementary School in Boston, the scene of an experiment in using computers across the curriculum. There, students work with Logo programming and computer controlled Lego construction materials. The work is both frequent enough (at least an hour a day) and open-ended enough for differences in styles to emerge.

When working with Lego materials and motors, most children make something move by attaching wheels to a motor that makes them turn. They see the wheels and motor through abstract concepts of what they are for: the wheels roll, the motor turns. Alex goes a different route. He looks at the objects concretely, without the filter of abstractions. He turns the Lego wheels on their sides to make flat shoes for his robot and harnesses one of the motor's most tangible features: the fact that it vibrates. When a machine vibrates it tends to travel, something normally to be avoided. When Alex runs into this phenomenon, his response is to make his robot (stabilized by its flat "wheel shoes") vibrate and thus move forward. When Alex programs in Logo he likes to keep things similarly concrete.

Learners are usually introduced to Logo programming through the "turtle," an icon on a computer screen that can be commanded to move around the screen and leave a trace as it goes. So, for example, the turtle can be told to move forward a hundred steps and turn ninety degrees with the commands FORWARD 100 RIGHT 90. Four such commands would have the turtle drawing a square. Programming occurs when a set of commands such as REPEAT 4 [FORWARD 100 RIGHT 90] are defined as a procedure: TO SQUARE. Alternatively, a subprocedure TO SIDE might be defined and repeated four times.

Alex wants to draw a skeleton. Structured programming views a computer program as a hierarchical sequence. Thus a structured program TO DRAW SKELETON might be made up of four subprocedures: TO HEAD, TO BODY, TO ARMS, TO LEGS, just as TO SQUARE could be built up from repetitions of a subprocedure TO SIDE. Alex rebels against dividing his skeleton program into subprocedures; his program inserts bones one by one, marking the places for insertion with repetitions of instructions. One of the reasons often given for using subprocedures is economy in the number of instructions. Alex explains that doing it his way was "worth the extra typing" because the phrase repetition gave him a "better sense of where I am in the pattern" of the program. He had considered the structured approach, but prefers his own style for aesthetic reasons: "It has rhythm," he says. In his opinion, using subprocedures for parts of the skeleton is too arbitrary, preemptive, and abstract. "It makes you decide how to divide up the body and perhaps you would change your mind about what goes together with what. Like, I might decide to think about the two hands together instead of each hand with the arms."[17]

[17] In its ideal, the structured method would have the programmer go beyond subprocedures to make one procedure that could be given different parameters to produce arms and legs, right and left sides, even differently shaped people. This aesthetic, known as "procedural abstraction," wants to see a right arm and a left leg

In his own way, Alex has resisted the pressure to believe the general superior to the specific or the abstract superior to the concrete. For Alex, thinking about hands as a subset of arms is too far away from the reality of real hands, just as taking a motor that was most striking as a vibrating machine and using it to turn wheels in the standard fashion was too far away from the real motor he had before him. While the structured programmer starts with a clear plan defined in abstract terms, Alex lets the product emerge through a negotiation between himself and his material.

Anne, also nine years old, is another bricoleur programmer. Her favorite hobby is painting, and she has become expert at using sprites in programs that produce striking visual effects.[18] A sprite is a second Logo icon, a turtle that can be set in motion. Once you give a sprite a speed and a heading, it moves with that state of uniform motion until something is done to change it, just like an object obeying Newton's first law.

In one of Anne's programs, a flock of birds (each bird built with a sprite) flies through the sky, disappears over the horizon, and reappears some other place and time. If all the birds were red, then it would be easy to make them disappear and reappear. The command SETCOLOR :INVISIBLE would get rid of them and SETCOLOR :RED would make them reappear. Anne, however, wants the birds to have different colors, and so making the birds reappear with their original color is more complicated.

One method for achieving this end calls for an algebraic style of thinking: you make the program store each bird's original color as the value of a variable, then you change all colors to invisible and recall the appropriate variable when the bird is to reappear. Anne knows how to use the algorithmic method, but prefers one that allows her to turn programming into the manipulation of familiar objects. As Anne programs, she uses analogies with traditional art materials. When you want to hide something on a canvas, you paint it out, you cover it over with something that looks like the background. Anne uses this technique to solve her programming problem. She lets each bird keep its color, but she makes her

disappear into a generalized abstract idea of "limb." For someone like Alex, the top priority is staying in touch with the concrete. He is aware of the importance of organizing his program in order to find his way around it, but he does so by giving it what he calls "rhythm" rather than a hierarchical structure of procedures and subprocedures.

[18] Anne's program has the merit of showing in compact form a set of qualities characteristic of the bricoleur, but usually more diffusely represented. For a more detailed account, see Turkle, *The Second Self*, 110–15.

program hide it by placing a screen over it. Anne designs a sprite that will screen the bird when she doesn't want it seen, a sky-colored screen that makes the bird disappear. Anne is programming a computer, but she is thinking like a painter.

Thinking like a painter does not prevent Anne from contributing a significant technical innovation to her fourth grade computer class. She is familiar with the idea of using two sprites to form a compound object. Her classmates and teachers have always done this by putting the sprites side by side. Anne's program is like theirs in using two sprites, one for the screen, one for the bird, but she places the sprites on top of each other so that they occupy the same space. Instead of thinking of compound objects as a way of getting a picture to be bigger, she thinks of compound objects as a way of getting sprites to exhibit a greater complexity of behavior, an altogether more subtle concept.

Thus, Anne's level of technical expertise is as dazzling in its manipulation of ideas as in its visual effects. She has become familiar with the idea of data structures by inventing a new one—her screened bird. She has learned her way around a set of mathematical ideas through manipulating angles, shapes, rates, and coordinates in her program. As a bricoleur, her path into this technical knowledge is not through structural design, but through the pleasures of letting effects emerge.

As in the case of Alex, Anne does not write her program in "sections" that are assembled into a product. She makes a simple working program and shapes it gradually by successive modifications. She starts with a single black bird. She makes it fly. She gives it color. Each step is a small modification to a working program that she has in hand. If a change does not work, she undoes it with another small change. She "sculpts." At each stage of the process, she has a fully working program, not a part but a version of the final product.

Anne is perfectly capable of producing a program with well-delineated subprocedures, although her way of creating them has little in common with the planner's approach.[19] Devotees of structured programming would frown on Anne's style. From their point of view, Anne should design a computational object (e.g., her bird)

[19] Bricolage does not exclude the use of subprocedures; it simply does not give their a priori delineation the status of a privileged method. For example, a part of a holistically conceived program can be demarcated as a subprocedure at any stage of programming. Subprocedures need not be "black boxes"; they, too, can be developed as the program grows as a whole. Indeed, the bricoleur may use as subprocedures programs that happen to be "lying around," possibly even programs that were originally made for very different purposes.

with all the required qualities built into it. She should specify, in advance, what signals will cause her bird to change color, disappear, reappear, and fly. One could then forget about "how the bird works"; it would be a black box. Anne's work dramatizes the feature of bricolage that was so salient for Lisa and Robin: the desire for transparency. Structured programmers usually do not feel comfortable with a construct until it is thoroughly black-boxed, with both its inner workings and all traces of the perhaps messy process of its construction hidden from view. Many such programmers feel a sense of power when they use black-boxed programs, perhaps because of the thought that others might take them up exactly as frozen.

Yet black boxing makes other programmers nervous rather than exultant. Anne did not want to package her constructs into opaque containers. Like Lisa and Robin, she enjoys keeping open the possibility of renegotiating their exact form. This means staying in touch with that form at all times. The bricoleurs in our study tend to prefer the transparent style, planners the opaque, but the program's authorship is a critical variable in this preference. Planners want to bring their own programs to a point where they can be black-boxed and made opaque, while bricoleurs prefer to keep them transparent; but when dealing with programs made by others, the situation is reversed. Now, the bricoleurs are happy to get to know a new object by interacting with it, learning about it through its behavior the way you would learn about a person, while the planners usually find this intolerable. The planners' more analytic approach demands knowing how the program works before interacting with it. They demand the assurance that comes from transparent understanding, from dissection and demonstration.

Despite the dominant ideology of the computer culture that privileges the structured, hierarchical, planner's style, Anne's case makes it clear that the difference between planners and bricoleurs is not in quality of product; it is in the process of creating it. In describing bricoleur programmers, we have made analogies to cooks and painters. Bricoleurs are also like writers who do not use an outline but start with one idea, associate to another, and find a connection with a third. In the end, an essay "grown" through negotiation and association is not necessarily any less elegant or easy to read than one filled in from an outline, just as the final program produced by a bricoleur can be as elegant and organized as one written with the top-down approach.

Do programmers graduate from bricolage when they develop greater expertise? Will Anne become a structured programmer in junior high? Our observations suggest that with experience, brico-

leurs reap the benefits of their long explorations, so that they may appear more "decisive" and like planners when they program on familiar terrain. Also, of course, they get better at "faking it." Still, the negotiating style resurfaces when they confront something challenging or are asked to try something new. Bricolage is a way to organize work. It is not a stage in a progression to a superior form. Interviews with computer scientists and their graduate students turned up highly skilled bricoleurs, most of them aware that their style was "countercultural." Indeed, there is a culture of programming virtuosos, the hacker culture, that would recognize many elements of the bricolage style as their own.

Within feminist scholarship there is a substantial body of literature that challenges the notion that human reason best expresses itself within terms of Western male gender norms.[20] For example, Carol Gilligan's work on moral reasoning calls into question the idea of one privileged, mature way of thinking. Gilligan's description of diverse approaches to moral reasoning is analogous to our contrast between the formal, canonical approach to programming and the concrete style of the bricoleur.[21] In the first, justice is like a mathematical principle: to solve a problem you set up the right algorithm, the right black box, you crank the handle, and the answer comes out.[22] In the second, a contextualized argument is like a concrete argument, one needs to stay in touch with the inner workings of the arguments, with the relationships and possible shifting alliances of a group of actors whose interests need to be negotiated.

Despite Anne's high level of achievement, theorists of structured programming would criticize her style for the same kind of reasons that a stage theorist of moral development would classify the most impressively articulate contextual thinker at a lower intellectual level than his or her "formal" colleague. In both cases, criticism would center on the fact that neither is prepared to take the final step to abstraction. Gilligan challenges this standard hierarchy; she uses her observations of moral reasoning through concrete situations to reject Lawrence Kohlberg's stage theory with

[20] See, e.g., Feyeraband; Hanson; Wittgenstein; and Feynman.

[21] Gilligan (n. 9 above). For a critical discussion of Gilligan's proposals and her reply see Linda K. Kerber et al., "On *In A Different Voice:* An Interdisciplinary Forum," *Signs: Journal of Women in Culture and Society* 11, no. 2 (Winter 1986): 304–33. Its methodological criticisms of Gilligan's treatment of the relationship between "voice" and gender do not detract from how her subjects illustrate the way of thinking we call "bricolage."

[22] It is an eleven-year-old, Jake, who when confronted with a moral dilemma describes it as "sort of like a math problem with humans" (see Gilligan, 26).

its determinate end point to development, an end point in abstract, universal principles.[23] If one branch of the development of moral reasoning moves toward the primacy of "justice," of the formal and analytic, Gilligan insists on equal respect for a different branch of development which leads toward increasingly sophisticated ways of thinking about morality in concrete terms of care through relationship and connection.

Gilligan is concerned with both morality and epistemology when she says: "The moral problem [for women] arises from conflicting responsibilities rather than from competing rights and requires for its resolution a mode of thinking that is contextual and narrative rather than formal and abstract."[24] Her language expresses a primary concern with the character of the morality, which, as she says, requires a certain mode of thinking. This emphasis on the character of the morality (rather than mode of thinking) is even more marked in recent writing where she redescribes Kohlberg's theory as being about only one side of moral reasoning. In this view, Kohlberg is talking about justice, thus leaving the other side of morality, namely, care, to her.[25]

This compromise, which splits off the *content* of moral judgments from the *mode of thinking* about them, blunts the force of Gilligan's observations as a challenge to something more general than moral reasoning; but that challenge is central to our argument. Kohlberg's theory of the development of moral judgment mirrors Piaget's theory of the development of intelligence per se. Both express the value-laden perspective on intellectual growth that has dominated Western philosophy. Piaget sees a progression from egocentric beginnings to a final, "formal stage" when propositional logic and the hypothetico-deductive method "liberate" intelligence from the need for concrete situations to mediate thinking.[26] In this vision, mature thinking is abstract thinking. We disagree: for

[23] Gilligan, 30 ff. Kohlberg had already been challenged on other grounds (see, e.g., John Gibbs, "Kohlberg's Stages of Moral Judgment: A Constructive Critique," *Harvard Education Review* 47, no. 4 [February 1977]: 43–61). Similar issues have been raised in critiques of Jean Piaget (see, e.g., Steven Toulmin, *Human Understanding* [Princeton, N.J.: Princeton University Press, 1972]). Toulmin argues that Piaget's experimental investigations reflect an a priori commitment to a Kantian position. We single out Toulmin because unlike most of Piaget's critics he does not quarrel with the detail of how the stages are described but with the epistemological assertion of the final end point.

[24] Gilligan, 19.

[25] Carol Gilligan and Jane Attanucci, "Two Moral Orientations," in *Mapping the Moral Domain,* ed. Carol Gilligan, Janie Victoria Ward, and Jill McClean Taylor (Cambridge, Mass.: Harvard University Press, 1990).

[26] Piaget and Inhelder (n. 4 above).

us, formal reasoning is not a stage, but a style. Gilligan's materials on the countercultural style of moral reasoning, like the countercultural style in programming, challenges the existence of hierarchical stages: for although Piaget would place the "concrete" Anne squarely in the preformal stage, her level of achievement undermines his assumptions about the superiority of the analytic and formal.

Thus, observation of programmers at work calls into question deeply entrenched assumptions about the classification and value of different ways of knowing. It provides examples of the validity and power of concrete thinking in situations that are traditionally assumed to demand the abstract. It supports a perspective that encourages looking for psychological and intellectual development within rather than beyond the concrete and suggests the need for closer investigation of the diversity of ways in which the mind can think with objects rather than the rules of logic.

Objects

Sooner or later in building objects with Lego, students at the Hennigan School where we met Alex run into the need for gears.[27] Looking at their work provides a good example of alternate styles applied to working with the same problem, formal styles that use rules and concrete styles that use objects.

The motors in the construction set turn at a high speed with low torque. A car built by attaching these motors directly to the wheels will go very fast, but will be so underpowered that the slightest slope of obstruction will cause it to stall. The solution to the problem with Lego cars is the same as that adopted by designers of real cars: use gears. Yet in order to use them effectively, children need to understand something about gear ratios.

If a small gear drives a larger gear, the larger gear will turn more slowly and with greater torque. It is the relative and not the absolute size of the two gears that counts. But when we interview children, we find that some of them reason as if the size of only one gear matters, as if they were following a set of rules such as "large gears are slow and strong" and "small gears are fast and weak."

[27] These experiments with Lego and programming are undertaken in a Piagetian spirit (see Jean Piaget, *La prise de conscience* [Paris: Presses Universitaires de France, 1951] for experiments that deal with how mechanisms work). For a personal statement about the power of gears as an introduction to formal systems, see Seymour Papert, "The Gears of My Childhood," in his *Mindstorms: Children, Computers, and Powerful Ideas* (New York: Basic, 1980).

Without the notion of relative size, such rules fail. Other children, and in our study, predominantly the girls, are less articulate and more physical in their explanations. They squirm and twist their bodies as they try to explain how they figure things out; and they get the right answer.[28]

Theorists who look at intellectual development as the acquisition of increasingly sophisticated rules would say that children run into problems if the rules they have built are not yet good enough.[29] The idea of "closeness to objects" enables us to consider a different kind of theory. Our observations suggest that the children who did so well did not have better rules, but a tendency to see things in terms of relationships rather than properties, access to a style of reasoning that allowed them to imagine themselves "inside the system." They used a relationship to the gears to help them think through a problem.

This "reasoning from within" may not be adequate for all problems about gears, but for the kind of problem encountered by the children in our project, it was not only adequate, but much less prone to the errors produced by a too simple set of rules. Relational thinking puts you at an advantage: you do not suffer disaster if the rule is not exactly right.

We have defined bricolage as a style of organizing work that invites descriptions such as negotiational rather than planned in advance, what Warren McCulloch called "heterarchical" rather than hierarchical.[30] The story of the children and their gears serves to introduce another characteristic displayed by many bricoleur programmers. We call this characteristic proximality or closeness to the object. There is little distance between Anne and her computational objects. Like the children, who "reasoned from within" with the gears, Anne psychologically places herself in the same space as the sprites. She experiences her screens and birds as tangible, sensuous, and tactile. She is down there, in with the sprites, playing with them like objects in a collage. When she talks about them her gestures with hand and body show her moving with and among them. When she speaks of them she uses language such as "I move here."

[28] Our sample does not allow us to say that girls did systematically better than boys. Research is in progress on this point. Our present discussion is about styles of explanation (rule driven vs. body syntonic) not distribution of abilities.

[29] For example, most of those inspired by the Carnegie-Mellon schools of artificial intelligence. See Ryszard S. Michalski et al., eds., *Machine Learning: An Artificial Intelligence Approach* (Los Altos, Calif.: Morgan Kaufmann, 1983).

[30] Warren McCulloch, *Embodiments of Mind* (Cambridge, Mass.: MIT Press, 1988).

The object relations school of psychoanalysis focuses on the way development progresses by a process of internalization of the things and people of the world. They come to live within us; they become the objects with which we think.[31] When psychoanalysts talk about "objects" they usually mean people.[32] Here we extend the idea of internalized "objects to think with" to the domain of everyday relationships with artifacts. It is not enough to ask whether individuals "like" or "don't like" to program because that puts the question on too high a level of generalization. "Liking" to program depends on forging a personally meaningful relationship with a computational object, a relationship that "fits." In forging this relationship, there are several dimensions of choice. People can choose among computational objects. For example, in the version of Logo used by Anne there was a choice between sprites and turtles. Some prefer the turtle, its static nature, the fineness in the way it draws. For others, these same qualities are reasons to reject the turtle as constraining, even unpleasant. They prefer the sprites, which move with flash and speed.

People can (and do) choose different ways of approaching the same object. Computational objects, like turtles and sprites, stand on the boundary between the physical and the abstract. You can see them, move them, put one on top of another. Yet, they are mathematical constructions. Canonical programmers treat a sprite more like an abstract entity, a Newtonian particle, while bricoleur programmers treat it more like a physical object, a dab of paint or a cardboard cut-out.

Computational objects offer a great deal to those whose approach to knowledge requires a close relationship to an object experienced as tactile and concrete. Some people are comfortable with mathematical exercises that manipulate symbols on quadrille-ruled paper. For many others, computational objects offer a physical path of access to the world of formal systems. For them, the ambivalent nature of computational objects may make possible a first access to mathematics.[33]

[31] For an excellent overview of the object relations perspective, see Jay R. Greenberg and Stephen A. Mitchell, *Object Relations in Psychoanalytic Theory* (Cambridge, Mass.: Harvard University Press, 1983).

[32] In contrast, D. W. Winnicott has some suggestive ideas about the power of the "transitional object"—the baby's blanket, the teddy bear—that in developmental terms mediates between experience of self and non-self. In the current context, it suggests the power of the inanimate in inner life (see D. W. Winnicott, *Playing and Reality* [New York: Basic, 1971]).

[33] The Logo turtle was designed to be "body syntonic," to allow users to put themselves in its place. When children learn to program in Logo, they are encouraged to work out their programs by "playing turtle." The classic example of

Feminist critics have related the standard notion of scientific objectivity to the social construction of gender: objectivity in the sense of distancing the self from the object of study is culturally constructed as male, just as male is culturally constructed as distanced and objective. From this point of view, Anne's proximal style is countercultural, reminiscent of Keller's description of geneticist Barbara McClintock's intimate relationship to the objects of her scientific study. For McClintock, the practice of science was essentially a conversation with her materials. The more she worked with neurospora chromosomes (so small that others had been unable to identify them), "the bigger [they] got, and when I was really working with them I wasn't outside, I was down there. I was part of the system. I actually felt as if I were right down there and these were my friends. . . . As you look at these things, they become part of you and you forget yourself."[34]

Alex and Anne relate to computational objects much as McClintock related to chromosomes, as does successful computer science graduate student Lorraine, who explains how she uses "thinking about what the program feels like inside" to break through difficult problems. "For appearances sake," she wants to "look like I'm doing what everyone else is doing, but I'm doing that with only a small part of my mind. The rest of me is imagining what the components feel like. It's like doing my pottery." This is in sharp contrast to programmers in the structured, canonical style who use their favorite device of black-boxing as a way to maintain distance. The idea of the black box, designed not to be touched, mediates between the structured (planning) style of organizing work and their relationship to computational objects. Structured programmers are not *among* the sprites, they act *on* the sprites.

The contemptuous comment of one fourth-grade boy who overheard a classmate talking about "being a sprite" when he programs can be interpreted from this point of view. "That's baby talk," he said. "I am not in the computer. I'm just making things happen there." The remark reflects an insistence on boundaries and the development of a worldview that will fall easily into line with the

---

this is developing the Logo program for drawing a circle. This is difficult if you search for it by analytic means (you will need to find a differential equation), but easy if you put yourself in the turtle's place and pace it out. (The turtle makes a circle by going forward a little and turning a little, going forward a little and turning a little, etc.) Turtles are a path into mathematics for people whose surest route is through the body (see Papert, *Mindstorms*).

[34] Keller, *A Feeling for the Organism* (n. 7 above), 117. Keller describes McClintock's approach as dependent on a capacity to "forget herself," immerse herself in observation, and "hear what the material has to say" (198).

canonical, objective science whose gender-based meanings Keller has delineated.

In our research we find a close relationship between bricolage, a style of organizing work, and proximality, a style of relating to the objects of work.[35] Our data are consistent with a model of styles as clusters of characteristics in which bricolage and proximality form the nucleus of one cluster ("concrete thinking") and planning and distality the nucleus of the other ("formal thinking"). These clusters are ideal types: our contention is not that the attributes in each cluster are exactly correlated but that each has internal coherency in the way that a stable culture is coherent.

So, for example, closeness to objects tends to support a concrete style of reasoning, a preference for using objects to think with, and a bias against the abstract formulas that maintain reason at a distance from its objects. Conversely, a distanced relationship with objects supports an analytic, rule- and plan-oriented style. Our theoretical conjecture is that degree of closeness to objects has developmental primacy; it comes first. The child forms either a proximal or a distant relationship to the world of things. The tendency to use the abstract and analytic or the concrete and negotiational style of thinking follows.

Although closeness to objects favors contextual and associational styles of work, it does not exclude the possibility of using a hierarchical one. Planning is not always an expression of personal style. It can be acquired as a skill, sometimes because it is needed to get a particular job done, sometimes as a facade to hide rather than express individuality.

Indeed, our data suggest that we may be underestimating the degree of association between proximality and bricolage. Some people adopt elements of the canonical style because they feel a social pressure to do so. In order to attract less negative attention, Lisa said that she decided to be a different kind of person, that is, more of a "planner." Robin says she "fakes it" and forces herself to black-box. Lorraine affects the discourse of a distanced style while

[35] In the seventy cases on which we report here, forty grade school children and thirty college students, we found these two dimensions of approach to programming in all but nine cases. Thus, empirically, we sometimes find each aspect of the concrete approach—bricolage as a style of organization and closeness to the object—without the presence of the other. In particular, one finds people who are planners but who enjoy a close relationship with concrete objects (and who experience computational objects this way). On the pairing of planning and what they call an interactive style with the computer, see Rosamund Sutherland and Celia Hoyles, "Gender Perspectives on Logo Programming in the Mathematics Curriculum," in *Girls and Computers*, ed. Celia Hoyles, Bedford Way Papers, 34 (London: University of London, Institute of Education, 1988).

something very different is going on in her head. Some bricoleurs respond to the dominant ethos of the computer culture by entering into an inauthentic relationship with the computer. This can lead to a paradoxical reaction: frustrated bricoleurs appear at first sight to be extremely rigid "planners." Some turn to a "cookbook" approach—like when in third grade, we were told to divide fractions by turning "the second fraction" upside down. When denied a chance to do their "real thinking," they turn to rules that do not require them to think at all. People like Lisa and Robin "escape to conformity," a reaction that muffles the manifestation of their distinctive voices in computing. Nevertheless, those voices are there. Recall the graduate student Lorraine, who says she tries "to look like I'm doing what everyone else is doing," in order to preserve "appearances." Her style is hidden beneath her efforts to fit in.

In our culture, the structured, plan-oriented, abstract thinkers do not only share a style but constitute an epistemological elite. We have never seen a case in which someone claims to have felt pressure to move away from the canonical style. Thus, since the phenomenon of "faking it" goes only in one direction, we conclude that true occurrence of the bricoleur/proximal combination are even more common than our raw count.

Another attribute associated with proximality and bricolage when working with computers is a tendency to anthropomorphize, to refer to the system as though it had human qualities. The anthropomorphization extends from the computational objects ("That sprite doesn't want to do what I tell it now") to the computer itself. Anne, for instance, has no doubt that computers have psychologies: "they think," she says "but can't really have emotions." She believes, however, that the computer has preferences, "He would like it if you did a pretty program." When it comes to technical things, Anne assumes the computer has an aesthetic: "I don't know if he would rather have the program be very complicated or very simple." Anne knows that the computer is just a machine, but she sees it nonetheless as a male companion, if only a limited one. Anthropomorphization, both of the computer system and its parts, does not follow from lack of technical expertise. It is a stylistic preference.[36]

[36] Anthropomorphization is not limited to children. It is a habit of mind shared by adults, including technical experts. When we say the computer "moves the queen" in a game of chess, the program has invited us to speak of it as though it had intentions. Programs within a computer system interact with each other in a way that supports models of the computer as composed of "agents" in communication. Computer scientists talk about a concept such as recursion with anthropomorphic

Very young children are in fact uncertain whether computers should be counted as alive or not alive, and argue the question hotly, debating the computer's aliveness on the basis of its psychology, its intentions, consciousness, and feelings. By age ten, most are sure that the computer is not actually alive. However, at this point, some children, like Anne, continue to behave with and talk about the computer as if it were sentient. They brag that it is helping them or complain that it is not. In this, they are not showing confusion about biology. They do not think that the computer is alive the way an animal is, rather that it has a "kind of life," the kind of life appropriate to a computer: it thinks.[37]

Others have a very different reaction. Once they are no longer perplexed by whether the machine might actually be alive biologically, they shy away from anthropomorphization. When they complain about the computer, they do so in objective terms: it is too slow, it does not have enough memory. Talking about the computer usually means talking about technical details.

Lise Motherwell, a researcher at the Hennigan school, did an intensive investigation of eight fifth-grade students. Motherwell found she could describe children's stances toward the anthropomorphization of the computer by distinguishing two styles: relational and environmental. Relational children, like those we are calling proximal thinkers, treat the computer as much like a person as they can, while environmental children, analogous to those we describe as preferring a distanced approach, treat it like a thing.

Once they have placed the computer in the not-alive category, the environmental children tend to settle with relief into treating it as a thing. This helps them to appropriate it through a relationship that involves distance, objectivity, and control. The relational children, once having settled the question of biological aliveness, get more comfortable with the machine by making it an interactive partner. In the computer they have found something in the domain of formal systems to which they can relate with informality. Three out of the four girls in Motherwell's study were relational; three out of the four boys environmental.[38]

In Motherwell's study, as in the study of children and gears, gender seems implicated in, but not a definitive influence on, style, consistent with observations of adult computer cultures where

metaphors: one agent "calls up" another, "wakes up" another, and "passes on a job." They sometimes even refer to the agents within a computer system as citizens of a "society of mind" (see Marvin Minsky, *Society of Mind* [New York: Simon & Schuster, 1987]; and Papert, *Mindstorms* [n. 27 above]).

[37] Turkle, *The Second Self* (n. 1 above), esp. chap. 1.

[38] Motherwell (n. 1 above).

some men are alienated from the dominant engineering style and many women work creatively within it. Again, as in our examples of Anne, Alex, Lisa, Robin, and Lorraine, the concrete style did not imply a lower quality of work. Concrete, proximal gear builders did just as well and in some cases better than the formal thinkers; children who anthropomorphize the computer are no less technically sophisticated than those who do not. The degree of concrete, proximal, and anthropomorphic thinking reflects not expertise but a preferred approach to knowledge.

Gender, closeness, and conflict

Several intellectual perspectives suggest that women would feel more comfortable with a relational, interactive, and connected approach to objects, and men with a more distanced stance, planning, commanding, and imposing principles on them.[39] Indeed, we have found that many women do have a preference for attachment and relationship with computers and computational objects as a means of access to formal systems. Yet in our culture computers are associated with a construction of science that stresses aggression, domination, and competition. The cultural construction of science leads to a conflict that considerably complicates our story of how women appropriate technology. In the case of computation, this conflict is particularly acute.

From its very foundations, science has defined its way of knowing in a gender-based language. Francis Bacon's image of the (male) scientist putting the (female) nature "on the rack," underscores the way objectivity has been constructed not only in terms of the distance of the knower from nature but also in terms of an aggressive relationship toward it (or rather toward her). From its very foundations, objectivity in science has been engaged with the

[39] Gilligan's work illuminates the relationship between bricolage and gender and Keller speaks to the gender meanings of the proximal style of relating to objects, be they physical objects such as gears or chromosomes or conceptual objects such as the elements of programming. A psychoanalytic perspective would place the roots of such approaches at an early stage of child development. If, in our culture, women are the primary caretakers for children, the earliest and most compelling experiences of merging are with the mother; differentiation and delineation take on gender meanings (see, e.g., Nancy Chodorow, *The Reproduction of Mothering: Psychoanalysis and the Sociology of Gender* [Berkeley and Los Angeles: University of California Press, 1978]; and Keller, *Reflections on Gender and Science* [n. 9 above]). The traces of such early experiences are culturally reinforced by continuing gender divisions of parenting roles and by the very different socialization of men and women.

language of power, not only over nature but over people and organizations as well. Such associations have spread beyond professional scientific communities; aggression has become part of a widespread cultural understanding of what it means to behave in a scientific way. Its methods are expected to involve "demolishing" an argument and "knocking it down" to size. Here the object of the blows is not a female nature but a male scientific opponent. Science is first a rape, then a duel.[40]

The traditional discourse of computation has not been exempt from these connotations. Programs and operating systems are "crashed" and "killed." We write this paper on a computer whose operating system asks if it should "abort" an instruction it cannot "execute." In our ethnographic studies of the social worlds that grow up around computing, we have found that this is a style of discourse that few women fail to note. Thus, women are too often faced with the not necessarily conscious choice of putting themselves at odds either with the cultural associations of the technology or with the cultural constructions of being a woman.

When Lisa had a confrontation with her instructor about the proper way to program, the computer culture and its canonical epistemology were represented by a person in authority with whom she argued. In other cases the tension comes from fears of what people might think rather than a confrontation with what someone actually thinks. Lorraine, who programs by imagining "what the components feel like," ends her description of her programming style by adding: "Keep this anonymous. I know it sounds stupid." In both these reactions there is a tension between the individual and an outside agency, but the conflict is internalized: the computer culture alienates by putting one in conflict with oneself.

When Lisa first found herself doing well in her programming course, she found it "scary" because she felt she needed to protect herself from the idea of "being a computer science type." In high school, Lisa saw young men around her turning to computers as a way to avoid people: "They took the computers and made a world apart." Lisa describes herself as "turning off" her natural abilities in mathematics that would have led her to the computer. "I didn't care if I was good at it. I wanted to work in worlds where languages had moods and connected you with people." Although Robin had gone through most of her life as a musician practicing piano eight

[40] Evelyn Fox Keller, "Baconian Science: The Arts of Mastery and Obedience," in *Reflections on Gender and Science*, 33 ff.; Carolyn Merchant, *The Death of Nature* (New York: Harper & Row, 1980); Donna Haraway, "The Biological Enterprise: Sex, Mind, and Profit from Human Engineering to Sociobiology," *Radical History Review* 20 (Spring/Summer 1979): 206–37.

hours a day, she, too, had fears about "guys who established relationships" with the computer. "To me, it sounds gross to talk about establishing a relationship with the computer. I don't like establishing relationships with machines. Relationships are for people."

In the vehemence with which many women insist on the computer's neutrality, on its being nothing more than a mere tool, there may be something more subtle going on than a clash between culture and personal style—a clash between personal style and sense of self. Many women may be fighting *against* having a close relationship to a computer or to computational objects. For some, like Lorraine, there is a clash because they want to belong to the dominant computer culture. But for others, the experience of closeness to the object is a source of conflict with themselves.

Lisa, like Anne, placed herself in the space of the computational objects she worked with, and she tended to anthropomorphize, responding to the computer as though it had (at least) an intellectual personality. In Lisa's case, her own style came to offend her because it had led her to what she experienced as a too close relationship with a machine. When Lisa began programming she saw herself as communicating with the computer, but the metaphor soon distressed her. "The computer isn't a living being and when I think about communicating with it, well that's wrong. There's a certain amount of feeling involved in the idea of communication and I was looking for that from the computer." She looked for it and she frightened herself. "It was horrible. I was becoming involved with a thing. I identified with how the computer was going through things."

In our research we find that women express such sentiments with particular urgency. We observe that a conflict fuels their convictions. In many cases, they are most comfortable with a style of thinking in which they get close to the objects of thought. The computer offers them such objects, but the closer they get to them the more anxious they feel. One remedy for their anxiety is denial. The more these people become involved with the computer, the more they insist that it is *only* a neutral tool. Again, their assertion is belied by the vehemence with which it is expressed.

Lisa's conflict with her instructor would be resolved in principle by a greater tolerance for her way of thinking; but addressing internal conflicts about being close to computers requires more than tolerance. It requires profound changes in the culture that surrounds the computer. For instance, if the computer is a tool, and of course it is, is it more like a hammer or more like a harpsichord?

The musician Robin is not distressed by her close relationship with her piano, which is also a machine. Lisa, who finds attachment

to the computer "unnatural," is not upset by her passion for the beautiful, heavy antique ink pens with which she writes. If Lisa had been in music school, it is most likely that she, like Robin, would not experience as threatening her sense of communicating with her instrument or her emotional involvement with it. Music students live in a culture that over time has slowly grown a language and models for close relationships with music machines. The harpsichord, like the visual artist's pencils, brushes, and paints, is a tool, and yet we understand that artists' encounters with these can (and indeed, will most probably) be close, sensuous, and relational. Indeed, the best artists will develop highly personal styles of working with them.

The development of a new computer culture would require more than environments where there is permission to work with highly personal approaches. It would require a new social construction of the computer, with a new set of intellectual and emotional values more like those applied to harpsichords than hammers.[41] Since, increasingly, computers are the tools people use to write, to design, to play with ideas and shapes and images, they should be addressed with a language that reflects the full range of human experiences and abilities. Changes in this direction would necessitate the reconstruction of our cultural assumptions about formal logic as the "law of thought." This point brings us full circle to where we began, with the assertion that epistemological pluralism is a necessary condition for a more inclusive computer culture.

Roadblocks and openings for change

While the computer supports epistemological pluralism, the computer culture has not. In some ways, though, the computer culture is catching up with the potential of the computer. In particular, significant openings for change within the technological culture come from a new emphasis on computational objects that is making itself felt in domains as diverse as debates about which personal computers are the best and how to build artificial brains.

Its simplest manifestation is the fashion for using icons to control personal computers. In a traditional IBM-style computer system control is through typing instructions. In an iconic system, the same effects are achieved by moving screen symbols. The current technology for the act of moving something on a screen falls

[41] On values for a new computer culture, see Seymour Papert, "Technological Thinking versus Computer Criticism," *Educational Researcher* 16, no. 1 (January 1987): 22–30.

short of what the computer industry expects to provide quite soon, but existing systems, such as the Macintosh's "mouse" and touch sensitive screens, already give a tactile sense that recalls Anne's experience of programming as collage.

Even superficial use of icons is enough to transform the perception of the computer by people who are using it in computationally simple ways. For example, it is commonplace knowledge that many writers who began to use computers reluctantly, as a necessary evil, now find they have friendlier relationships mediated by the icons, the mouse, and the cozier appearance of a Macintosh. Although these particular warmer relationships do not involve programming, their influence may mean that the next generation of people like Lisa and Robin will come to programming courses with a different sense of who "owns" the computer.

A multiplicity of technical methods does not by itself lead to pluralism. It can simply lead to competition, to which the computer industry is scarcely averse; neither are those computer users who seem to enjoy conversations that engage them in heated debate about the merits of their favorite system. It is only when we understand the computer as a projective screen for different approaches to knowledge that we can listen to these conversations as a striving for pluralism. Different people are comfortable with different systems. When people fight about the IBM versus the Macintosh, what they may be trying to do is defend their intellectual styles. Yet, the debate is cast as an effort to prove the other side wrong, as if it would be impossible to prove both sides right.

A multiplicity of technical methods can also lead to elitism. The Logo language allowed Anne and Alex to program in their own ways, but in many educational settings Logo is defined as the computer language for children who have not reached the top stage in Piaget's hierarchy. In such settings even as sophisticated a thinker as Anne or as creative a thinker as Alex would get their liberty at the cost of having their intellects defined as immature. Similarly, the very success of the Macintosh has often been cast in terms that reflect the elitism of the dominant computer culture. The Macintosh iconic interface has been marketed as "the computer for the rest of us," with the implication that the rest of us need things made simple and do not want to be bothered with technical things.

As it happens, the popularity of icons may be settling this argument, a conclusion that has many interpretations. The designers of computer interfaces might interpret it as final proof of the technical superiority of icons. A psychologist might read it as a challenge to the concrete/formal split: perhaps most people are concrete thinkers most of the time, and formal thinking is used for

acceptibility or prestige or functionality. Others might simply say that icons are "easier." Taken individually these positions do not support pluralism. What is important here is whether the support for icons is part of a larger shift toward an acceptance of concrete ways of thinking.

If the shift goes far it may be through the connection of the icons to something deeper, a philosophy of "object oriented programming."[42] In the traditional concept of a program, the unit of thought is an instruction to the computer to do something. In object-oriented programming the unit of thought is creating and modifying interactive agents within a program for which the natural metaphors are biological and social rather than algebraic. The elements of the program interact as would actors on a stage. This style of programming is not only more congenial to those who favor concrete approaches, but it also puts an intellectual value on a way of thinking that is resonant with their own. In principle it could undermine the canonical position in at least two ways: first, within the world of programming through legitimating alternative methods; second, in the larger intellectual culture, by supporting trends in cognitive theory that challenge the traditional canon.

Until recently, prevailing models of cognitive theory have bolstered the commitment of psychologists and educators to the superiority of formal, or at least formalizable, thinking. They were given support by the cognitive theorists most influential in the computer world, the leaders of the artificial intelligence (AI) community. In the late 1970s and early 1980s, the model of AI with the greatest visibility was the rule-based "expert system," with its model of mind as a structured information processor. Critics of how computers influence the way we think cited the information-processing model as demonstrating the instrumental reason and the lack of ambiguity allegedly inherent in all computational thinking about intelligence.[43] Artificial intelligence is not a unitary enterprise, however, and recently, another model has become increasingly prominent: "emergent AI."[44]

[42] For a nontechnical discussion, see Alan Kay, "Microelectronics and the Personal Computer," *Scientific American* 237 (September 1977): 230–44, and "Software's Second Act," *Science* 85 (November 1985): 122 ff.

[43] See, e.g., Hubert Dreyfus, *What Computers Can't Do: The Limits of Artificial Intelligence,* 2d ed. (New York: Harper & Row, 1979); and Joseph Weizenbaum, *Computer Power and Human Reason* (San Francisco: W. H. Freeman, 1976).

[44] The reaction within artificial intelligence against abstract, propositional, rule-driven methods was given literary expression in the writings of Douglas Hofstadter (see, e.g., "Waking Up from the Boolean Dream, or Subcognition as Computation," in his *Metamagical Themas: Questing for the Essence of Mind and Pattern* [New York: Basic, 1985], 631–65). Two other manifestations of this reaction

Emergent AI does not suggest that the computer be given rules to follow but, rather, it tries to set up a system of independent elements within a computer from whose interactions intelligence is expected to emerge. Its sustaining images are drawn not from the logical but from the biological and social. Families of neuronlike entities or societies of anthropomorphized subminds and sub-subminds are in simultaneous interaction from which mindlike process is expected to emerge. These models are sometimes theorized in notions of "mind as society," where negotiational processes are placed at the heart of all thinking. Those who espouse and support such models are far more inclined to find bricolage acceptable than are classical Piagetians. What concerns us here is not making value judgments about these trends in AI, just as we are not advocating a choice between the use of icons and the use of textual instructions in computer operating systems. What does concern us is that the new trends—icons, object-oriented programming, actor languages, society of mind, emergent AI—all create an intellectual climate in the computer world that undermines the idea that formal methods are the only legitimate methods.

Thus, recent technological developments, in interfaces, programming philosophy, and artificial intelligence, have created an opening for epistemological pluralism. We began by presenting the notion of epistemological pluralism in reference to three streams of thought—feminism, ethnography of science, and psychology—which, although different in many ways, converge in reasserting the importance of objects in thinking. We close by noting the opportunity for new alliances between them and computation.

Louis Althusser writes about psychoanalysis that the important breakthrough was not any particular statement about the mind, but the step of recognizing the unconscious as an object of study that defines a new theoretical enterprise.[45] Psychology long had considered the rational and the conscious as the quintessential mental activity; Sigmund Freud shifted the ground to the irrational and the unconscious. The unconscious was not given recognition as only an important factor but, rather, became an object of science in its own

are Minsky (n. 36 above); and David E. Rumelhart, James L. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (Cambridge, Mass.: MIT Press, 1986). For our more extended comments on the "two AIs," see Seymour Papert, "One AI or Many," *Daedalus* 117, no. 1 (Winter 1988): 1–13; and Sherry Turkle, "Artificial Intelligence and Psychoanalysis," *Daedalus* 117, no. 1 (Winter 1988): 241–68.

[45] Louis Althusser, "Freud et Lacan," *La Nouvelle Critique*, nos. 161–62 (December–January 1964–65), 88–108.

right. Similarly, we imagine the emergence of a science of thought that would recognize the concrete as its central object.

There is every reason to think that the revaluation of the concrete will open the computer culture to accepting the computer as an expressive medium that encourages distinctive and varied styles of use. There is every reason to think that this pluralistic computer culture could be more welcoming and nurturing to women and to men. Gilligan has said that women can protect the recognition "of the continuing importance of attachment in human life."[46] The evidence from our research on programming styles leads us to conclude with an analogous speculation. Feminist scholarship could make a crucial contribution to the (until now) predominantly male computer culture by promoting recognition of the diverse ways that people think about and appropriate formal systems and by encouraging the acceptance of our profound human connection with our tools.

Program in Science, Technology, and Society (Turkle)
Media Laboratory (Papert)
Massachusetts Institute of Technology

[46] Gilligan (n. 9 above), 23.