Computer as Rorschach

Sherry Turkle

When we think about the social roles of the computer we tend to think of a familiar list of administrative, financial, and governmental functions and a set of social, political, and legal problems that are raised by what computers do. In this essay we look at social roles that the computer plays, not as a direct result of what they do, but because of the relationships people form with computers: how people think about computers, how they use computers to think and to not think about other things, how deeply subjective private worlds of computation (including preferences for different styles of programming) and a medium in which people work through personal and political concerns that are far from any instrumental use of the computer. In sum, we shall be looking at the computer as a metaphor and as a projective medium, and suggesting that this subjective side of the computer presence is highly relevent to understanding issues concerning computation and public life.

There is an extraordinary range of textures, tones, and emotional intensity in the way people relate to computers—from seeming computer addiction to confessed computerphobia. I have recently been conducting an ethnographic investigation of the relationships that people form with computers and with each other in the social worlds that grow up around the machines. In my interviews with people in very different computing environments, I have been impressed by the fact that when people talk about computers they are often using them to talk about other things as well. In the general public, a discourse about computers can carry feelings about public life—anxieties about not feeling safe in a society that is perceived as too complex, a sense of alienation from politics and public institutions. Ideas about computers can also express feelings about more private matters, even reflecting concerns about which the individual does not seem fully aware. When we turn from the general public to the computer experts, we find similar phenomena in more developed forms. There, too, ideas about computers carry feelings about political and personal issues. But in addition, the expert enters into relationships with computers which can give concreteness and coherence to political and private concerns far removed from the world of computation. In particular, the act of programming can be an expressive activity for working through personal issues relating to control and mastery.

Of course, among technologies, the computer is not alone in its ability to evoke strong feelings, carry personal meaning, and create a rich expressive environment for the individual. People develop intense and complex relationships with cars, motorbikes, pinball machines, stereos, and ham radios. If computers are an exception to the general rule that there is a subjective side to people's relationships with technology, it is insofar as they raise this commonly known phenomenon to a higher power, and give it new form as well as new degree.

Other technologies, knives for example, can serve as projective screens: do we associate them with butter or with blood? But we can come close to having people agree that before it is a part of eating or killing, a knife is a physical object with a sharp edge. We shall see that the elusiveness of computational process and of simple descriptions of the computer's essential nature undermine such consensus and make the computer an exemplary "constructed object," a cultural object which different people and groups of people can apprehend with very different descriptions and invest with very different attributes. Ideas about computers become easily charged with multiple meanings. In sum people often have stronger feelings about computers than they know.

The Subjective Computer

A ticket agent who uses computers to make airline reservations begins a conversation about the computer by presenting it as a totally neutral object—programmed, passive, completely under the control of its operators and their input, threatening only in its impersonality—and then moves on in the same conversation to descriptions of the machine as a presence in which the line between person and thing seems nearly to dissolve. When confronted in a conversation by the possibility of computers which might serve as psychotherapists, judges, or physicians—that is, whose functions would be ones which are now seen as quintessentially "human"—many people react with a force of feeling by which they themselves are surprised. Some people try to neutralize feelings of discomfort by denying that such things as intelligent computers could exist outside of science fiction, but then try to buttress their arguments by adding in unabashed self-contradiction that while such things may be possible, they "ought not be allowed to happen." In talking about computers, people often make implicit reference to two scenarios that have long been explicit in science fiction

0147-2011/80/0115-0006$03.50/1

plots: computers might change something about the way people think, and computers might develop minds of their own. In the complexity of our responses to the idea of machine intelligence, we see an expression of our stake in maintaining a clear line between the human and the artificial, between what has consciousness and autonomy and what does not, between a notion of "mechanical" calculation and of "human" judgment and emotion. The fact that the computer touches on a sphere—intelligence—that man has long thought to be uniquely his, means that even popular discourse about computers can raise tense questions about what is man and what is machine.

Questions like this are posed, if only implicitly, by our everyday use of language; that is, by our use of computational metaphors. In our culture, the fact that there is talk about such things as repression, the unconscious, and the superego, influences the way in which people think about their problems, their pasts, and their possibilities, even for people who do not "believe in" psychoanalysis. In the case of psychoanalysis, technical ideas were taken up as powerful metaphors by a nontechnical public and used as building blocks in a discourse about politics, education, and the self; that is, as building blocks in the development of a psychoanalytic culture. These ideas took many shapes and turned up in many different places as they became integrated into advice to the lovelorn as well as into theories of psychology. Computers, too, introduce a world of new language to those who work with and around them. And since this language is about cognitive processes that often seem at least superficially analogous to those which go on in people, this language is brought into everyday vocabulary.

Students speak of "dumping core" when they are asked to spill back course contents during an exam. Engineers complain of being "stuck in a loop" when problem solving is difficult and all paths lead to dead ends. A travel brochure for a condominium village in Hawaii assures the reader that a stay in Wailea means a sure addition to his "fond memory bank." Today's language for thinking about thinking is growing richer in computational metaphors. When we say that we have an idea that needs to be "debugged," we are referring to a computational model of dealing with global complexity through local intervention. When a computer scientist refuses to be interrupted during an excited after-dinner conversation and explains that he needs to "clear his buffer," he is using an image of his mental terrain in which access to interactive processing capacity can be blocked by a buffer zone that must be empty before it can be crossed.

We do not yet know whether these metaphors, commonly dismissed as "manners of speaking," are having an effect on the way we think about ourselves, perhaps by effecting an unconscious transfer between our ideas about machines and our ideas about people. In academic psychology, however, such transfer has become explicit: in the mid-1950s, the presence of the computer, a complex material embodiment of cognitive functions, gave American psychologists a new model for thinking about cognition, one which stressed the need to posit complex internal processes in order to understand even simple behavior (something that traditional behaviorism, in its attempts to avoid theorizing about internal states and processes, had declared outside the realm of good science). For example, behaviorists had spoken of the behavior of "remembering," but computational models reintroduced the notion of "a memory" into general psychology. Today, computational and information-processing models seem on their way to becoming the new dominant paradigm in psychology and have made serious inroads in other behavioral and social sciences. It is a plausible conjecture that, as in the case of psychoanalysis, today's technical computational language will filter into tomorrow's popular language.

Some might imagine that such subjective aspects of the computer presence and the use of computation for model building are either a private matter, of concern to the individual involved, or of interest to the theoretical psychologist, but without any bearing on issues of public concern except insofar as *misinformation* about computers can obscure discussion of public problems. In fact, the situation is more complex. We can observe the "subjective computer" in the language and the projections of individuals, but it does impact on the collectivity. There are several ways in which it can influence our approach to issues concerning computers in public life. First, when the computer acts as a projective screen for other social and political concerns, it can act as a smokescreen as well, drawing attention away from the underlying issues and onto debate "for or against computers." Second, feelings about computers (often largely projective in origin) can become formalized into "ideologies" of computer use, that is, into beliefs about what the computer can, will, and should do. These powerful computer ideologies can decrease our sensitivity to the technology's limitations and dangers as well as blind us to some of its positive social possibilities. And finally, along with the "constructed computer" comes the social construction of computer expertise.

When a school wishes to purchase a computer system, whom shall they consult? There are at least a hundred thousand Americans who have bought small, personal computers for their homes. Many are parents—it is natural that the school and the PTA should look to them as experts. And from a purely technical point of view, many of them are. But we shall see that the relationship of many computer hobbyists to the computer carries a vision of what is important in computation that systematically leaves some things out. Other "expert" groups introduce different biases.

The general public tends to think of a computer expert as defined by purely technical criteria but, in fact, computer experts are often distinguished from each other by subjective stances (such as an emotional feeling about what is important about computation) as well as by their technical capabilities. Even people who are extremely sensitive to the way in which personal preoccupations and

political preferences can masquerade as "neutral" expertise in other technical fields often think that in computer science, things are different. One popular image is that computer expertise is a neutral quantity that can be acquired like a piece of hardware and be relied upon to perform in a steady and reliable way. It is as though people tend to see computer experts (often referred to as "computer people") as being "like computers." But different relationships with computers, different aesthetics of how to use them and what they are good for, structure computational value systems whose implications extend far beyond the technical. Even preferences among styles of programming can have a politics. One programming aesthetic puts a premium on having all elements of the program "on the table" and available to the programmer as "primitives." With so many little pieces, each one has to be made as small as possible to get them to all fit into the workspace, and so the criteria of elegance for this "flat" style of work are associated with highly condensed programming at the bottom level. Before the recent plummeting in the cost of memory, this ground floor condensation allowed economies of memory space that made its elegance highly cost effective. But because the structural building blocks are small, condensed, and numerous, modifications are virtually impossible without changing the whole system. An alternative aesthetic (top down, structured programming) builds up hierarchical programs using large, internally unmodifiable modular blocks. This often uses more memory but allows easy modifications with less reliance on a master programmer. The system is socially desirable, but some programmers find it constraining, unaesthetic, "good for organization, but bad for the artist." We shall return to the question of computational values and politics. Here I want only to suggest that understanding different subjective relationships with computation may be necessary to understanding and evaluating the views of computer experts on issues of public policy such as what kind of computer system needs to be built in a given situation. Indeed, such understanding may be a necessary step towards a kind of computer literacy that prepares the citizen to make responsible political judgments.

Computer as Smokescreen

Computational metaphors are only one element in the construction of a new, highly charged, and often highly self-contradictory popular discourse about computers. There is the everyday reality: the average American meets computer power when he makes a telephone call, uses a credit card, books a motel room, goes to the bank, borrows a library book, or passes through the checkout counter of the local supermarket. There is the science-fiction surrealism: the computer of the future is presented as threatening (HAL in *2001*), all-knowing (the "Star Trek" computers), and all-powerful (in the movie *Demon Seed*, a computer succeeds in impregnating a woman, resulting in a computer-human baby). And there is media image making, as television, popular journalism, advertisements, even games and toys, bombard us with an extraordinary range of images about what the computer really is and what it might be. The computer is portrayed as supercalculator, superenemy, superfriend, supertoy, supersecretary, and in the case of the bionic people who populate television serials and children's imaginations, computer as a path into a future of supermen and women.

Computers are portrayed as good and bad, as agents of change and of stagnation. Talking about computers and money, computers and education, computers and the home, evokes tension, irritation, anticipation, excitement. Some of the intensity reflects the schizophrenic splitting in the images of computers in our culture. But some of it comes from the use of the computer as a projective screen for other concerns. And although we do not yet know if computational metaphors and ideas about computers are changing the way we think, it is already clear that our popular and highly projective discourse about computers can discourage us from thinking things through.

Consider, for example, the problem of how computers make it easier to violate the privacy of the individual through the automatic accumulation of data about him. Traditional notions of the right to privacy are challenged when most social transactions leave an electronic trace. The computer presence has made the problem of privacy more urgent and visible. More attention is being devoted to it because decisions about its protection can no longer be postponed when there is the prospect of their being "hardwired" into national information systems. But all too often, discussions of computers and privacy focus on the computer. This draws attention off the fact that organizations violated citizen privacy long before there were computers to help them do the job. And attention is drawn off the fact that the root of this serious problem lies not in our computer systems but in our social organization and political commitments and that its solution must be searched for in the realm of political choice, not of fancier technology. On the issue of privacy, the computer presence could serve to underscore an underlying problem; instead, talk about the computer serves as its smokescreen.

A similar smokescreen effect is present in the following images of the computer, all of them comments made by computer science professionals at a recent symposium on computers and society.

> *The computer that constrains*:
> "You get on an elevator and you're wearing a badge in a particular office building and you try to go to the fifth floor. The computer in the elevator says, 'No, that's not your floor.'"
>
> *The computer that encourages violence*:
> "A group of students were standing around a console playing Space War and I heard one student say to the other: 'Don't you think we should get more points for killing than for merely surviving?' It was a perfectly reasonable statement in that context, and

I'm afraid it may turn out unhappily to become a slogan for the era of the home computer."

The computer that atrophies the mind:
"Now that we're using calculators and no longer multiply in our heads, we may find an almost epidemic rise in things like dyslexia, learning disabilities, inability to work, a propensity to industrial accidents and auto accidents."

Let us consider the third image: the computer that atrophies the mind. Later discussion made it clear that the speaker who prophesied that the calculator age meant an increase in learning disabilities was deeply concerned about the contemporary crisis in education, where functional illiteracy after a high-school career has become commonplace. But our understanding of that crisis is not advanced if concern about a falling educational standard is expressed as complaints about calculators that may disenable multiplication neurons. The other images carry similar dangers. The fact that computer games are violent, like the fact that television programs are violent, makes a statement not about technology but about our society. The most disturbing thing about the student's comment about the game of Space War has nothing to do with computers. Its language is not very different than that which was used while our government was fighting and justifying the war in Vietnam, to take only one example. That we now see it reflected back to us on television and CRT screens is a comment not on the computer presence but on the internalized violence of our society.

Sophisticated information systems do facilitate increased surveillance of individuals, and shoot 'em down video games on personal computers can multiply the images of violence that enter our livingrooms. This tendency of computers to increase the urgency of many problems could in principle give rise to sharper social criticism. But in practice this seldom happens. Complaints about *computers* invading privacy and about *computer* games being violent are daily used to short-circuit discussion of political responsibility and the banalization of violence. Behind our conversations about the computer that constrains us (the computer that "won't let you off on the fifth floor," or, as in another common example, "won't let you change your airline reservation") is often our sense of having limited access to what we want to see and understand. There are people and large organizations behind the "computer" that constrains. When people's sense of political limitation is translated into statements about technology, about computers "hiding things from us," political discussion has been neutralized and the possibility for appropriate action has been subverted.

It is easy to catalogue the interests (industrial, governmental) that tend to be served when political choices are represented as technical problems. These interests exert forces from without. But other forces, harder to catalogue, also encourage this same kind of obfuscation. In a certain sense, these come from within. If a memory or a dream disturbs our sense of who we are as individuals, we "forget" it—we make it unconscious. As a society, we also find ways to "forget" the collectively unacceptable. Comfortable and habitual inactions are threatened by serious talk about such matters as how the decisions of large corporations affect our political and biological environment or about the consequences of gross inequalities of resources and power. We develop a paradoxical language for talking about such matters that allows us to forget the real issues. And one of the most powerful of these languages is technical. The strategy is not new, and insofar as the computer has a role here, it is to provide new means towards already familiar ends. But the new means make a difference.

People are particularly willing to embrace the computer as a technical explanation for things that might otherwise raise disturbing questions. Consider the situation of the airline clerk we met earlier. She frequently finds herself confronted by the anger of clients whose reservations cannot be honored. Her standard excuse is "Our computer fouled up." Like workers in a thousand other bureaucracies all over the world, the airline clerk need never call the organizational policies of her company into question. She need never call her employer into question because the computer is there to blame. It is felt by her to be an autonomous entity (it can act with agency) and so it is *blamable*, yet it is not a fellow worker to whom she would feel bonds of loyalty. This permits her a conscience-calming collusion with the client without jeopardizing her security as a "company person." What is it about the computer that makes it such an effective actor in situations like this? In order to answer this question we need to step back and try to understand people's tendency to anthropomorphize computers. Most particularly, we need to appreciate that it is deeply rooted in the nature of the computer itself. It does not necessarily reflect a lack of information or naive beliefs in the "intelligence of machines," either present or future. Many who find the anthropomorphization of computers offensive would like to make it go away by educating the public to understand "what computers really are." But they miss an epistemological issue: computation is irreducible. We can know more and more about it, but we never come to a point where we can completely define it in terms of more familiar things.

The computer theorist, like other scientists, sets up a conceptual frame of reference within which he works, and defines the computable within this framework. But even then, what he has isolated as the computable, the "essential computer," presents no easy analogies with other objects in the world (as the airplane does the bird)—except, of course, and this is a point to which we shall return, for its analogies with people. To explore this further requires that we proceed by a kind of paradox. We try to understand the epistemological isolation of the computer by looking at some of the many ways in which people try to projectively relate it to other things, each valid within a particular horizon, although many are inconsistent with each other.

Computer as Rorschach

The computer's capacity as a projective device resembles that of the Rorschach, perhaps the best known and most powerful of psychology's projective measures. In the Rorschach, the individual is presented with an ambiguous stimulus, a set of inkblots. How he responds to them is a window onto his deeper concerns. And so it is with the computer. First, as in the case of the Rorschach, whose blots suggest many shapes but commit themselves to none, we have noted that the computer is difficult to capture by simple description. We can say that it is made of electrical circuits, but it does not have to be. A computer can be made (and several—for fun—have been) of tinkertoys, and quite serious computers have been made using fluidic rather than electrical circuits. Although airplanes can come in all shapes and can be described in all sorts of ways, there is no conceptual problem in stating their essential function: they fly. There is no equally elegant, compelling, or satisfying way of defining the computer. Of course, one could say that it computes, that it executes programs. But the execution of a program can be described on many levels: in terms of electronic events, machine language instructions, high-level language instructions, or through a structured diagram which represents the functioning of the program as a flow through a complex information system. There are no necessary one-to-one relationships between the elements on these levels of description, a feature of computation which has led philosophers of mind to see the computer's hardware-software interplay as highly evocative of the irreducible relationship between brain and mind. The irreducibility of the computer to other things encourages, indeed it even seems to coerce, its anthropomorphization. This is further reinforced by the computer's interactive properties (you type to it and it types back to you) and by the unpredictability of programs (although the programmer inputs all instructions, their interaction soon becomes sufficiently complex that one can seldom foresee the results of their operation).

Computers are certainly not the only machines that evoke anthropomorphization. We often talk about machines as though they were people: we complain that a car "wants to veer left." We even talk to machines as though they were people: we park a car on a slope and warn it to stay put. But usually, when we "talk to technology," we have a clear path in mind for transforming any voluntary actions we may have ascribed to a machine into unambiguously mechanical events. We know that friction on the wheels caused by the emergency brake will prevent gravity from pulling the car down the hill. But when we play chess with a computer and say that the computer "decided" to move the queen, it is much harder to translate this decision into physical terms. Of course, an engineer might well reply that "all the computer really does is add numbers." And indeed, in a certain sense, he is right. But thinking of the computer as adding does not get us very far towards understanding why the computer moved the queen. Saying that the computer decided to move the queen by adding is a little like saying Picasso created *Guernica* by painting. And there is more than a touch of irony in the engineer's trying to undermine the anthropomorphization of the computer by using what is ultimately an anthropomorphic image of adding.

The reaction to ELIZA, a conversational natural language program that simulated the responses of a Rogerian psychotherapist, threw people's tendency to attribute human characteristics to computers into sharp relief. By picking up on key words and phrases it had been programmed to recognize, the program was able to ask questions and make responses ("I AM SORRY TO HEAR YOU ARE DEPRESSED," "WHAT ELSE COMES TO MIND WHEN YOU THINK OF YOUR FATHER?") that made "sense" in its conversational context (a therapy session).

Most of those who originally had access to ELIZA knew and understood the limitations on the program's ability to know and understand. The program could recognize character strings, but it could not attribute meaning to its communications or those it received. And yet, according to its creator, Joseph Weizenbaum, and much to his consternation, the program seemed to draw some of them into closer relationships with it. People confided in the program, wanted to be alone with it, seemed to attribute empathy and understanding to it. In my conversations with students about their experiences with ELIZA, the personalization of the involvement with the program often seemed tied to the issue of predictability. Many referred to the feeling of being "let down" when the program became predictable. When they had cracked the code, when they knew which inputs would provoke which responses, when they knew which inputs would cause the program to become "confused," then "computer confidences" became boring.

People tend not to experience themselves or other people as completely predictable. When asked what it means to be a person and not a machine, most people use plain talk to describe what the more philosophically minded might call the "ineffable." Machines are most people's everyday metaphor for invoking predictability and, insofar as the computer is able to simulate the kind of unpredictability we associate with people, it threatens our concept of machine. Here is a machine that is not "mechanistic." Locally, it has mechanistic components, but seen globally, these disappear and you are dealing with a system that surprises.

Something else that makes analogies between the computer and mechanical antecedents (like adding machines) unsatisfactory is that computers can be programmed into autonomy from their human users. On the simplest level, after a few sessions of an introductory computer science course, the novice programmer knows how to write programs that would, in principle, go on forever, let us say, because step three is an instruction that says return to step one. Such programs will never stop; that is, until somebody "kills" them by pulling out the plug, turning off the

machine, or pressing a special control key on the computer terminal which is designed for just such moments. I interviewed a group of college students as they went through an introductory programming course and most could remember strong feelings about what one referred to as his first "forever program."

"Forever" is overwhelming because we can't know it or our place in it. Perhaps a "forever program" gives a glimpse, however ephemeral, of what it might mean. Such glimpses are rare, sometimes occasioned by looking at a mountain, or at a sunset, and are almost always accompanied by strong emotions. In the case of the iterative program, the image of "forever" is created by the programmer himself, perhaps intensifying its evocative power, its fascination.

The computer demonstration called the GAME OF LIFE has this evocative quality. The game begins with a checkerboard of dark and light cells in a given state; cells turn from dark to light and light to dark depending on the state (dark or light) of their immediate neighbors. Such local instructions produce a changing, evolving global pattern. Like a biological system, this computer program can generate global complexity out of local simplicity. The game fascinates, touching on our fascination with self-perpetuating systems, with generativity, and "forever." It also brings into focus a compelling tension between local simplicity and global complexity in the working of the computer. Locally, each step can be predicted from the step before. But the evolution of the global pattern is not graspable. This play between simplicity and complexity is among those things that makes computation a powerful medium for the expression of issues related to control. And perhaps it is in the range of programming relationships that this projective potential is maximally realized.

Programming as Projective

Depending on how the programmer brings the computer's local simplicity and global complexity into focus, he will have a particular experience of the machine as controlled or controlling. Both levels are there; people display different patterns of selective attention to each of them and end up with different relationships to control and power in their programming work. Out of this range of relationships we will look at two very different ones. We see a first style in X, an ex-programmer, now a university professor who describes himself as "having been a computer hacker." X experiences his computer power as a kind of wizardry. Wizards use spells, a powerful local magic. X's magic was local too. He described his "hacker's" approach to any problem as a search for the "quick and dirty fix" and described his longterm fantasy that he could walk up to any program, however complex, and "fix it, bend it to my will." As he described his intervention, he imitated the kind of hand gestures that a stage magician makes towards the hat before he pulls out the rabbit.

X's involvement was in a struggle with the program's complexity—what was most gripping for him was being on the edge between winning and losing. He described his hacking as walking a narrow line: make a local fix, stay aware of its potential to provoke unpredicted change or crash the system, test each system's flexibility to the limit. For X, the narrow line has "holding power." Stories of weekends at the terminal with little to eat and little or no rest were common, as were reflections on not being able to leave the terminal when debugging a program clearly required getting some sleep and looking at the whole in the morning instead of trying to "fix it" by looking at it line by line all night. For X it was his style of programming that led him to identify with what was for him a computer "subculture," that of the hacker. His process of identification seemed analogous to that of a creative independent virtuoso who recognizes his peers not by the "job" they have nor by their academic credentials, but because they share his sense of the personal importance, the urgency of creating in the medium in which they work.

Many hackers have dropped out of academic programs in computer science in order to devote themselves exclusively to computers. Based neither on a formal job nor on a research agenda, the coherency of the hacker subculture follows from a relationship with the "subjective computer"; that is, with a set of values, a computational aesthetic, and from a relationship with programming that may be characterized as devotion to it as a thing in itself. In university settings all over the country, where hackers are often "the master programmers" of large computer operating systems, academic computer scientists complain that the hackers are always "improving the system," making it more elegant according to their aesthetic, but also more dificult to use.

Some have characterized the hacker's relationship with computation as "compulsive," but its urgency can be otherwise described. The hacker grapples with a computational essence—the issue of how to exert control over global complexity by mastery of local simplicity. The mechanism embodied in the lines of code under his immediate scrutiny is always simple, determined, certain—but the whole constantly strains to escape the limit of his ability to "think of it all at once," to see the implications of his actions on the larger system. And this is precisely what he finds so exciting.

A second programmer, Y, is also a computer professional, a microprocessor engineer who works all day on the development of hardware for a large industrial data system. He has recently built a small computer system for his home and devotes much of his leisure time to programming it. Whereas for hacker X, the excitement of programming is that of a high-risk venture, Y likes it as a chance to be in complete control. Although Y works all day with computers, his building and programming them at home is not "more of the same." He experiences his relationship to the computer as completely different in the two settings. At work he describes himself as part of a whole that he cannot see and over which he feels no mastery or ownership: "At work what I do is part of a big

system; like they say, I'm a cog." At home he works on well-defined projects of his own choosing, projects whose beginning, middle, and end are all under his control. To him, the home projects seem a kind of compensation for the alienation of his job. He observes that he works most intensively on his home system when his tasks at work seem mostly a project of "somebody else having parceled things out . . ." and furthest away from any understanding of "how the whole thing fits together."

X and Y have very different senses about what is most satisfying about programming. These translate into different choices of projects, into different computational values, and ultimately into what we might call different computational aesthetics. X likes to work on large, "almost out of control" projects; Y likes to work on very precisely defined ones. X finds documentation a burdensome and unwelcome constraint. Y enjoys documentation; he likes to have a clear, unambiguous record of what he has mastered. Much of his sense of power over the program derives from its precise specifications and from his attempts to continually enlarge the sphere of the program's local simplicity. There is certainly no agreement between X and Y about what constitutes a "good" program or a "good" computer application.

X, like many other people I have spoken to who identify with the hacker subculture, sees business systems and IBM and its products (FORTRAN, COBOL, IBM timesharing, and IBM computers themselves) as particularly "ugly." A company like IBM is interested in system reliability, and this means trade-offs in the system's "plasticity." A hacker may complain that such systems hold back both the computer and the programmer. He is often more sympathetic to computer applications which touch on the area of Artificial Intelligence, the enterprise of programming computers to do things (like having vision, speech, and chess-playing ability) that are usually considered intelligent when done by people. His sympathy is not surprising. In Artificial Intelligence projects, the hacker can see an embodiment of his sense of what is most exciting about the computer—the way unpredictable and surprising complexity can emerge from clever local ideas. At the other extreme, programmer Y's commitment to computers is to what is most precise, predictable, and controllable. For Y, what is powerful about the computer is definitionally in a different realm than the human mind with its vagueness and unpredictability. He may rule Artificial Intelligence out of court because there is as yet no agreed upon specification of what it is to be "intelligent." ("How can you build something which has not been reduced to 'specs'?") For the hacker, this usually poses no problem. In fact, his sense of computational power is incompatible with "specs."

People bring computers into their homes for many different reasons, but questionnaire data on over a hundred computer "hobbyists" (here defined as people who have had a computer in their home for several years—that is, before the advent of mass marketed "turnkey" systems) and nearly 150 hours of follow-up interviews with 30 of them suggested that Y's style of dealing with the computer, his computational values and aesthetics, are widely shared in this group. Like Y, other hobbyists have built their computers from kits, and many continue to work as close to the machine as possible, preferring assembly language to higher level languages, and preferring to write their own assemblers even when commercial ones are easily available. The hobbyist's relationship with the computer he has worked on, often built "from scratch," and nearly always carefully documented, can be heavily invested with a desire for a kind of personal control that can be passed on to his children.

Although advertisements for personal computers have stressed that they are an investment in your child's education—that computers have programs that can teach algebra, physics, the conjugation of French verbs—hobbyists don't speak about the importance of giving their children a competitive advantage in French, but of a competitive advantage in "the computer." Most hobbyists feel that the stakes are high. They believe that computers will change politics, economics, and everyday life in the twenty-first century. Owning a piece of it, and having complete technical mastery over a piece of it, is owning a little bit of control over the future.

For many hobbyists with whom I spoke, the relationship with their home computer carries longings for a better and simpler life in a more transparent society. *CoEvolution Quarterly, Mother Earth News, Runner's World*, and *Byte* magazine lie together on hobbyists' coffee tables. Small computers become the focus of hopes of building cottage industries that will allow the hobbyist to work out of his home, have more personal autonomy, not have to punch a time card, and be able to spend more time with his family and out-of-doors.

Some see personal computers as a next step in the ecology movement: decentralized technology will mean less waste. Some see personal computers as a way for individuals to assert greater control over their children's educations, believing that computerized curricula will soon offer children better educations at home than can be offered in today's schools. Some see personal computers as a path to a new populism: personal computer networks will allow citizens to band together to run decentralized schools, information resources, and local governments.

Many of the computer hobbyists I have interviewed talk about the computers in their livingrooms as windows onto a future where relationships with technology will be more direct, where people will understand how things work, and where dependence on big government, big corporations, and big machines will end. They represent the politics of this computer-rich future by generalizing from their special relationship to the technology, a relationship characterized by simplicity and a sense of control. In this tendency to generalization, they are not alone. People often take a particular way of relating to the computer; that is, they take their personal sense of what is important, interesting, and valuable about computers, and generalize it into beliefs about "computers in general." This process

of generalization and ideology formation can be rapid. I saw it begin with a group of 25 college students, computer "newcomers," whom I followed through their first computer science course. I spoke to them several times during the course about their reactions to learning about computers and programming: how did they see the computer, how did they feel about what they were learning.

Many of the students began the course with an image of the computer as a complex and powerful entity. But for some, with an elementary knowledge of the machine and of programming came a way of thinking about the computer that began to approach the view we have characterized as common to many hobbyists, a view of the computer as simple and controllable. ("The machine is dumb; just a giant calculator.") And for about half of the students, an image of a primitive computer whose power was based on the ability to perform arithmetic functions became their image for all of computation. In the process, their attention turned away from questions relating to the complexity of computation. They showed little interest in highly speculative issues about the future of Artificial Intelligence or in such down-to-earth problems of sloppily written complex systems that have gotten out of hand. Such systems (written, rewritten, and locally revised by different programmers, indeed by different teams of programmers through the years) can become a patchwork of local fixes, each with an inevitable, but often unknown, impact on the working of the whole. If you need to change such programs, the change can only take the form of yet another local fix, and the results of doing so are unpredictable. When we refer to such systems as incomprehensible, this does not mean that we cannot understand their local workings. It means that we cannot act on the program as though we understood it as a whole. We cannot know the consequences of our actions. The programs become autonomous in the sense that making changes to them becomes too "dangerous" to try. But because the students saw the programs they were writing in their course as easily modifiable, they could not really see how such problems could arise. Several dismissed the very possibility with the phrase "Garbage in—Garbage Out" (GIGO).

One might think that the problem here is in the nature of "introductory" material. In the teaching of chemistry, for example, we usually find that it makes most sense to begin with the simplest stuff, with the material that will give students the most confidence that they can make the subject "their own." And then when they move on from their high-school to their college chemistry classes, they are shown how to cast aside their high-school models of atoms as "wrong." Images of electron shells and precise orbits are replaced by models of orbitals, suborbitals, and probability densities. But in the case of the computer, things are different. The kind of programming that typically goes on in an introductory course encourages an emphasis on the "local simplicity" view of the computer. But at every level of expertise you can have a choice of focusing on simplicity or on complexity. There is no "truth" in the Rorschach inkblots that you finally see if you examine them long enough, and a particular set towards computation can be maintained at very different levels of expertise.

By the end of their one semester course, most students in my study had averaged seventy-five hours at the computer terminal. Many of the hobbyists I interviewed who had logged many thousands of hours and had completed some very complex projects were as solidly committed as the students to a view of the computer that focused on its local simplicity. And like the students, they, too, used their experience with the computer as a basis for dismissing issues that might emerge from computational complexity. It may well be that for some of them, their use of the computer as emblematic of the personally and politically controllable gave them strong reasons to want to hold on to this view. But that they were *able* to do so reflects something about the nature of computation. The view of computation as locally simple can be shared by programmers at very different levels of expertise because it is not technically wrong: all programs can be described locally, and *at least in principle* all programming goals can be achieved while retaining complete control over the system. *In practice*, many hobbyists are led by their passion for documentation to become masters of the art of local (most often line-by-line) description of programs and are led by the individualism of their computer culture into habits of work and choices of programming projects that reinforce a style of highly controlled programming. Thus a tradition, an aesthetic, and relationships both with people and machines maintain a sense of computation similar to that which tends to be encouraged by working with the small, tightly controlled programming projects typical of first courses.

The "blind spots" of those who invoked formulas like GIGO to dismiss the problem of incomprehensible programs went beyond the inability to see the consequences of computational complexity. The remark reflects a vision of programming as a technical act and as an individual act: if a program is incomprehensible, it is because someone wrote bad code. For an individual working alone that might be true, but it is a mistake to think about computation in other settings as an extension of the computation that one does in one's home or for a problem set.

Computation is a social act, the sum total of everything it takes to make a particular computational event occur: the hardware, the teams of people creating the necessary software, the organizations of people, bureaucracies, and industries in which it happens. The incomprehensibility of the large programs used by such organizations as the Internal Revenue Service can have a great deal to do with such social factors as the uncommitted relationship between the programmers and the organization, the structure and the instability of the programming teams, the way in which authority is delegated. Even the programming environment in which the work is done (what languages are used, what debugging systems are available, etc.) can depend on political choices within the organization. None

of these factors is intrinsic to computation. None of these factors is made apparent by extrapolating from most experiences of recreational or classroom computation.

Our discussion of "blind spots" and of programming experience helps us to bridge our earlier distinction between social problems that follow from what computers do and those that follow from how people think about computers. The social problems that arise from the presence of cumbersome, effectively unmodifiable programs in large organizations are in the class of problems raised by "what computers do." These problems may be compounded by difficulties in understanding their nature that are rooted in more subjective perceptions. The way in which we, as a society, deal with problems posed by what computers do is influenced by our ways of thinking about computers. The subjective side of computation is not without its objective consequences. The blind spots that I noticed among my sample of beginning students and hobbyists are only one example. We spoke about the computer "as Rorschach." But of course there is a difference. Unlike Rorschach blots, computers are also powerful social actors, and what people project onto them—these "socially constructed computers"—can themselves become social presences that influence policy makers, educators, engineers and the general public. In the last analysis, how people think about computers *is* "something that computers do."

Computer Literacy

The observation that the way people think about computers can exacerbate problems caused by what computers do leads easily to a standard response: educate them. There is an active movement of advocates and activists of "computer literacy," the minimum that everyone needs to know about computers in order to function effectively as a citizen. Schools and federal agencies, magazines and clubs, the computer industry, even the manufacturers of children's games are all entering the business of educating the public. There is no doubt that people are learning more about computers. But our glimpse of the subjective side of computation alerts us to some potential problems about what they may be learning. In most cases, and certainly in computer literacy courses that use curricula designed for grade school and adult education classes, people are learning simple programming skills and a set of "facts" about the computer. But we have seen that "facts" about the computer do not come in neutral information packets. Computer literacy is usually defined as knowledge about the computer. If we accept the idea of computation as a social act, it would be more appropriate to define it as knowledge about computers and people. In this essay, we have raised several issues that need to be taken into account by this kind of humanistic computer literacy movement.

We have seen that what seems like the obvious first step in computer education, learning to write small programs, can lead to a paradox. The computer educator hopes to give the student a more objective understanding, but the result can be to bias the student's perception of computation against recognizing phenomena associated with complexity. Several possible strategies have been suggested for dealing with this paradox. The student's model of computation might develop differently if his first computer experience was to modify a large pre-existing program rather than to create his own tiny ones. Explicit discussion of issues related to system complexity could be introduced into elementary computer education.

A second issue relates to selection. There clearly are different styles of relating to computers. The styles are so distinct that those who practice one are prone to see those who practice another as wrong, fuzzy headed, even bizarre. When speaking about programmers and their styles, we used the metaphor of subcultures. A standard computer literacy curriculum easily could become the vehicle which defines the "normal" and the "deviant" among these cultures. Any educationally "official" computer culture will encourage only some people to think of working with computation as being a good "fit" with who they are. When we think about computer education in the next decade, we are no longer talking about the education of a small group of people who will become computer specialists, computer experts. We are talking about computer literacy for the masses of people who will need to feel comfortable with computers in order to feel comfortable and unintimidated by daily life. The goal should be to give as many of them as possible the sense of belonging in a computer-rich society.

A third issue has to do with an unknown: how different styles of relating to computers may transfer to other things. We have noted that choosing a programming language and a programming style implies a cluster of cultural characteristics, values, ways of thinking. X's and Y's programming styles suggest strategies for dealing with problems that have nothing to do with the computer. There may be a transfer of some of these ways of thinking from computation to other things. If we acknowledge that a computer literacy program may be training in habits of thought, then it must be evaluated in these larger terms.

There is the fourth issue of anthropomorphization. The phenomenon makes many people uneasy. Some hope that objective knowledge about how computers "really" work will make it go away. But anthropomorphic imagery, supported by the computer's projective capacities, seems deeply embedded in the nature of computation. A responsible approach to computer education must take it more seriously, must understand its genesis and multiple functions, whether in the end it decides to oppose, exploit, or ignore it.

A fifth and final issue touches on the way in which the computer—as it becomes implicated in ways of thinking about politics, religion, psychology, and education—can raise challenging, even disturbing, questions for individuals. For example, in an introductory programming course, a college sophomore saw how seemingly intelligent and seemingly autonomous systems can run on programs. This led him to his first brush with the idea (which

others have first encountered via philosophy or psychoanalytic thought) that there might be something illusory in people's subjective sense of autonomy and conscious self-determination. Having seen this idea, he rejected it, with arguments about the irreducibility of man's conscious sense of himself that paralleled those of Freud's more hostile contemporaries both in their substance and in the emotion behind them. In doing so, he made explicit a commitment to a concept of man to which he had never before felt the need to pay conscious attention.

The reference to psychoanalysis brings us full circle to an analogy I made at the beginning of this article. There, I noted that twentieth-century popular culture has appropriated psychoanalytic ideas that were first developed in a technical context, and I conjectured that computational models for thinking about the mind might undergo a similar fate. Here I consider a very different aspect of the analogous relationship between computation and psychoanalysis—not how they can be similarly *accepted* but how they both carry messages which are likely to be resisted and *rejected*. Psychoanalytic notions of the unconscious, of infantile sexuality, and of Oedipal relationships provoked strong resistance before being accepted into either academic or popular cultures. Psychoanalysis is a framework for thinking—we might call it a "subversive science"—that challenges humanistic and "common sense" models of man as an autonomous agent. In doing so, it calls into question some of our taken-for-granted ways of thinking about ourselves. Computational frameworks share some of this "subversive" quality. They, too, provoke strong feeling. Opinions are divided: some people welcome computational analogies with people as the basis for a new kind of scientific humanism, while others warn that such models deny us that which is specifically human in our nature. Some embrace the prospect of Artificial Intelligence as an adventure for the human spirit, while others see much about the enterprise as obscene.

Although the phenomena around the "subjective computer" we have dealt with in this essay are highly visible in our culture, the groups that are most involved in computer education and the computer literacy movement tend to ignore them. Each has a different reason for doing so. The computer industry is committed to presenting computers as neutral technical objects that can enter daily life in a non-disruptive way. The personal computer magazines and hobbyist movement have a different motive for "normalization." Their effort is to assimilate everything to activities within the technical reach and the intellectual style of the owner of a very small system. Schools are intent on avoiding the controversial. They have had enough trouble with sharp debates over sex education and such experiments as the Man as a Course of Study program. They are willing to "take on" computation as a cost-effective adjunct to their standard curricula. It is not in their immediate interest to "see" other aspects of the computers they have taken on.

The leadership of industrial, recreational, and educational computing share a language for talking about computer education and computer literacy that is technical and instrumental and selectively ignores the more highly charged aspects of the computer presence. When these do come up, they tend to be denied as nonexistent, viewed as transitional phenomena that people need to be educated out of so that computers may more appropriately be seen as "just a tool." Of course the computer is a tool, but man has always been shaped by his artifacts. Man makes them but they in turn make him. In the case of the computer, we may confront a tool that can catalyze a change in how we think about ourselves; for example, by making us aware on a daily basis of being only one among many other possible forms of "rule driven" intelligence.

We spoke of the emergence in the mid-twentieth century of a psychoanalytic culture, a culture that had an influence on how people thought about their lives, about raising their children, and about the stability and instability of political systems. It is too soon to tell whether we are entering a computer culture that will have anything near this level of impact on us. But in our discussion of the subjective computer we began to see traces of such a culture in formation. There is the rapid spread of computational ideas into everyday language, there is the appropriation of information-processing models in psychology as well as in other behavioral and social sciences. If psychoanalytic ideas became culturally embedded through their embodiment in therapeutic practice, computational ideas are growing their own roots in education. There are cultures growing up around the computer that use the machines as metaphors for thinking about people and social organization. We wear a dangerous set of blinders if we do not appreciate and further explore how computers can become the carriers of culture and of a challenge to our way of thinking about ourselves. If nothing else, a fuller appreciation of this subjective side of the technology should lead us to a critical reexamination of what each of us takes for granted about "the computer" and to an attitude of healthy skepticism towards any who propose simple scenarios about the "impact of the computer on society."□

READINGS SUGGESTED BY THE AUTHOR:

Armer, Paul. "Attitudes Towards Intelligent Machines." In *Computers and Thought*, edited by Edward A. Feigenbaum and Julian Feldman. New York: McGraw Hill, 1963.

Neisser, Ulrich. "Computers as Tools and as Metaphors." In *The Social Impact of Cybernetics*, edited by Charles Dechert. Notre Dame, Indiana: The University of Notre Dame Press, 1966.

Weizenbaum, Joseph. *Computer Power and Human Reason: From Judgment to Calculation*. San Francisco: W.H. Freeman and Co., 1976.

Sherry Turkle is assistant professor of sociology at Massachusetts Institute of Technology, and author of Psychoanalytic Politics: Freud's French Revolution.