In *The Second Self*, Sherry Turkle looks at the computer not as a “tool,” but as part of our social and psychological lives; she looks beyond how we use computer games and spreadsheets to explore how the computer affects our awareness of ourselves, of one another, and of our relationship with the world. “Technology,” she writes, “catalyzes changes not only in what we do but in how we think.” First published in 1984, *The Second Self* is still essential reading as a primer in the psychology of computation. This twentieth anniversary edition allows us to reconsider two decades of computer culture — to (re)experience what was and is most novel in our new media culture and to view our own contemporary relationship with technology with fresh eyes. Turkle frames this classic work with a new introduction, a new epilogue, and extensive notes added to the original text.

Sherry Turkle is Abby Rockefeller Mauzé Professor of the Social Studies of Science and Technology at MIT and Founder and Director of the MIT Initiative on Technology and Self.

“Anyone who wishes to know about the effects of computers on American society today would do well to read *The Second Self*.” — Howard Gardner, *New York Times Book Review*

“Turkle has created an excursion into thought. . . . Sociologists, psychologists, and philosophers can benefit from examination of the principles put forth by Turkle.” — *Byte*

“A remarkably readable book that should appeal to anyone with the faintest interest in contemporary society and where it’s headed.” — *Newsday*
The Second Self: Computers and the Human Spirit
Children use the computer in their process of world and identity construction. They use it for the development of fundamental conceptual categories, as a medium for the practice of mastery, and as a malleable material for helping forge their sense of themselves. The computer is a particularly rich and varied tool for serving so wide a range of purposes. It enters into children’s process of becoming and into the development of their personalities and ways of looking at the world. It finds many points of attachment with the process of growing up. Children in a computer culture are touched by the technology in ways that set them apart from the generations that have come before.

Adults are more settled. In the worst of cases, they are locked into roles, afraid of the new, and protective of the familiar. Even when they are open to change, established ways of thinking act as a braking force on the continual questioning so characteristic of children. Family and work responsibilities and the very real constraints of social class can make it too risky to cast doubt on certainties. But there are events and objects that cause the taken-for-granted to be wrestled with anew. The computer is one of these provocations to reflection. Among a wide range of adults, getting involved with computers opens up long-closed questions. It can stimulate them to reconsider ideas about themselves and can provide a basis for thinking about large and puzzling philosophical issues.

Of course, this doesn't happen for everyone. Some people are intimidated by computers and keep their distance. Others see them merely “as a tool” and assimilate them into their nine-to-five life. But within the world of home computer owners, within the world of virtuoso programmers known as “hackers,” and within the world of artificial intelligence experts, a community dedicated to the enterprise of building “thinking machines” and computational theories of mind, people have taken up the computer in ways that signal the development of something new. The “something
new” takes many different forms. A relationship with a computer can influence people’s conceptions of themselves, their jobs, their relationships with other people, and with their ways of thinking about social processes. It can be the basis for new aesthetic values, new rituals, new philosophy, new cultural forms.1

In this chapter on personal computer owners I put the emphasis on these developments by focusing on the role of the computer as a catalyst of culture formation. What made the first generation of personal computer owners a culture was not only that they used the same hardware, read the same magazines, and attended similarly organized users’ groups, but also, as I learned, something else. They shared a quality of relationship with the computer, an aesthetic of using the computer for transparent understanding. In this chapter I show how, despite important changes in personal computers themselves, in the kinds of people who are buying them, and the uses to which they are being put, this issue of transparent understanding remains an important theme for a new generation’s relationship with their machines.

The Birth of a Personal Computer Culture

In 1975 the popular symbol of “the computer” was the IBM card, a fragile object—”Do not fold, spindle, or bend”—an object that was to be fed into a large machine owned by a bank, a corporation, a research institution. As this book goes to press, another image dominates public awareness: a person, intense and concentrated, hunched over the keyboard of a small computer with an interactive screen display.

In 1975 the most common complaints about the computer were about billing errors (“the computer did it”) and lost airplane reservations (“the computer is down”). The threat from the computer was the threat from the impersonal system that knew you only as a number. Now the threat seems all too personal, captured in the fear that a child or a spouse will get “hooked” and become addicted to a machine.

Nineteen seventy-five was a watershed year. Until then the number of people who kept a personally owned computer in their homes could not have exceeded a few score. Then quite suddenly things changed. A small company named MITS announced a kit for $420 containing everything one would need to build a small computer. It was called the Altair, and it started a revolution. Within a year thousands of people had bought the Altair or the spate of products that quickly followed to fill the growing demand for kits, for parts, and for ready-made small machines. Within a
year thousands of people had joined computer clubs and subscribed to personal computer magazines. Within five years the number had climbed into the millions.

From the beginning, most promotional literature and popular accounts of home computer use emphasized the instrumental: how computers could teach French or help with financial planning and taxes. But from the beginning it was clear that this utilitarian rhetoric was not the only source of excitement. I spoke at length to members of that first generation of personal computer owners, the people who bought and built small computers in the late 1970s.* Some justified their purchase of a personal computer by referring to a specific job—monitoring a home heating system, keeping records for a small business, establishing an inventory for a kitchen or a toolroom—but in most cases they also described a point at which their sense of engagement with the computer had shifted to the noninstrumental. They spoke about “cognitive play” and “puzzle solving,” about the “beauty of understanding a system at many levels of complexity.” They described what they did with their computers with phrases such as “building another room in my mind.” Once people actually had a computer in their home, the most interesting thing about it became the computer itself, not for what it might do, but for how it made them feel.

We saw that computers can broaden children’s sense of their talents and possibilities. Computers can do something of the same sort for adults who have more entrenched ideas about what they can and cannot do, their aptitudes and ineptitudes. Many people think of themselves as incapable of doing anything technical or mathematical, and learn from their interactions with a personal computer that this simply isn’t true.

* I began my study of personal computer owners in 1978 by sending a questionnaire survey to 300 New England home computer owners (their names had been randomly drawn from the roster of a home computer club and the subscription list of a personal computer magazine). Ninety-five responded to the survey. I conducted personal interviews with twenty-seven of these original respondents and in 1978–79, I interviewed another thirty-three home computer owners in Boston, New York, and San Francisco. By 1981 the technology available for home use had grown sophisticated and foolproof enough to attract people without technical backgrounds. I interviewed a second group of thirty East Coast home computer owners who had just bought their machines. In addition to the home computer owners with whom I had extended conversations (usually visiting them in their homes) I attended meetings of personal computer clubs and users’ group meetings. (Users’ groups are special associations that form within personal computer clubs that bring together the “users” of a particular kind of machine.)
One high-school French teacher who describes himself as “having a love affair with a TRS-80” explains: “After Sputnik, when I was in grade school and then in junior high, there was all that fuss, all the kids who were good in math got to be in special classes. Rockets were going up, men trying to go to the moon. Decisions about things. Scientists seemed to be in charge of all that.” For him, learning to program has a symbolic meaning, making him feel that he has crossed a line to become a member of a technical culture from which he had previously felt excluded. A more complex example of the computer changing one’s self-conception is illustrated by Barry, who has always thought of himself as a “technical person,” but who was “scared out of real science.”

Barry went to college for two years, hoping to be an engineer, then he dropped out and went to technical school. He has a job calibrating and repairing electronic equipment for a large research laboratory, and he likes his job because it gives him “a chance to work on a lot of different machines.” But he came to it with the feeling of having failed, of not being “analytic or theoretical”: “I always had a great deal of difficulty with mathematics in college, which is why I never became an engineer. I just could not seem to discipline my mind enough to break mathematics down to its component parts, and then put it all together. I could never grasp what was really important in science.” Five years before I met him, Barry bought a programmable calculator and started “fooling around with it and with numbers the way I had never been able to fool around before.” He says that “it seemed natural to start to work with computers as soon as I could.” To hear him tell it, numbers stopped being “theoretical,” they became concrete, practical, and playful, something at which he could tinker.

I’ll pick up the calculator, and if I don’t know how to do a problem I’ll play with the calculator a few minutes or a few hours and figure it out. It’s not so much that the calculator does a particular calculation, but you do so many, have so much contact with the numbers and the results and how it all comes out, that you start to see things differently. The numbers are in your fingers.

The calculator and the computer made numbers seem concrete. “They put mathematics in my hands and I’m good with my hands.”

Barry says that he doesn’t think of engineering as a pipe dream, nor does he think of what he is learning with the computer as something that could make it real. In terms of his career, nothing has changed, but in a more private sphere a great deal has changed. Working with the computer has made Barry reconsider himself.

I really couldn’t tell you what sort of things I’m going to be doing with my computer in six months. It used to be that I could tell you exactly what I would be
thinking about in six months. But the thing with this, with the computer, is that the deeper you get into it, there is no way an individual can say what he’ll be thinking in six months, what I’m going to be doing. But I honestly feel that it’s going to be great. And that’s one hell of a thing.

Barry’s world has always been divided into the people who think they know what they’ll be doing in six months and people who don’t. Barry has crossed this line, and now he has started to call other lines into question, the lines he has always used to “mark off” what was off-limits to him. In school, his inability to do the kind of mathematics he respected made him lose respect for himself as a student. The computer put mathematics into a form he could relate to. More important than the mathematics he has mastered, he has come to see himself as a learner.*

These issues recall those that came up when I discussed adolescents, identity, and computers. But Barry and the other home computer owners I shall be speaking of here are adults. They have the whole computer—the hardware and the electronics, as well as the program—to work with. And, most important, as adults they live in many worlds—worlds not just of play and learning, but of work and productivity. They bring new pressures and greater complexity to their relationships with their machines.

The computer is Janus-like—it has two faces. Marx spoke of a distinction between tools and machines.3 Tools are extensions of their users; machines impose their own rhythm, their rules, on the people who work with them, to the point where it is no longer clear who or what is being used. We work to the rhythms of machines—physical machines or the bureaucratic machinery of corporate structures, the “system.” We work at rhythms that we do not experience as our own. What is most striking in the story of the revolution that began with the Altair personal computer is that for many people the computer at home becomes a tool that compensates for the ravages of the machine at work.

Like Barry, most of those who bought or built the first generation of personal computers were “technical people.” And, like Barry, most worked in the electronics industry. Many of them were computer programmers. Unlike Barry, however, most were dissatisfied with their jobs.

* To some extent, hobbies have always played this kind of role. But the way people talk about buying a personal computer makes it clear that unlike the case for many other hobbies they have tried in the past—stereo, carpentry, model railroading—it makes them feel a part of something that is growing and that the society at large really cares about. It can mean new feelings of empowerment; it can mean crossing a frontier that separates tinkering from real technology.
In the course of the decades from the 1960s to 1980s programmers watched their opportunities to exercise their expertise in a spontaneous manner being taken away. Those who are old enough remember the time when things were different as a kind of golden age, an age when a programmer was a skilled artisan who was given a problem and asked to conceive of and craft a solution. For those who are young, the memory of such times remains alive in the collective mythology of the shop. Increasingly, programs are written on a kind of assembly line. The professional programmer works as part of a large team and is in touch with only a small part of the problem being worked on. Thus programmers are particularly sensitive to the fragmentation of knowledge and the lack of a feeling of wholeness in work to which so many of us are subject. And when these “proletarian programmers” buy home computers they bring this sensitivity to their hobby. When those I interviewed spoke of their home computers they spoke of the sense of power that came from having full knowledge of the system, of the “feeling of control when I work in a safe environment of my own creation.” They spoke of how good it felt to work with “whole problems.”

In many ways Carl is typical of the first generation of personal computer owners: thirty-five, college educated. He had worked as a programmer for over eleven years when his company “made the switch” to structured programming: “good for business, death for the joy of the work.” He has recently built a small computer system for his home, and he devotes much of his leisure to programming it. Although Carl works all day with computers, what he does with his home system is not more of the same.

Carl has thought a great deal about what he calls his “mental ecology.” At work he sees himself as only one small part of a process he cannot understand or control. “Like they say, I’m just a cog.” He thinks about the need to put different kinds of thinking in balance, and about how his home computer can help.

If you never or rarely get to finish things at work, if your job is basically to make little pieces and it’s somebody else’s job to make them fit into a whole, then working with a computer at home can give you an experience of getting it all together. You do the whole thing—building up from machine code to finished project. It makes you feel in balance.

**Personal Computers and Personal Politics**

When personal computers were first introduced they were most accessible to people like Carl whose work experiences prepared them to use the
machines (with the early models this took considerable technical expertise) and gave them a desire to exploit the machines’ potential for creating worlds of transparency and intelligibility. There is something else notable about the introduction of personal computers: they came on the scene at a time of dashed hopes for making politics open and participatory. Personal computers were small, individually owned, and when linked through networks over telephone lines they could be used to bring people together. Everything was in place for the development of a politically charged culture around them.⁶

The computer clubs that sprang up all over the country were imbued with excitement not only about the computers themselves, but about new kinds of social relationships people believed would follow in their wake. Of course there was talk about new hardware, new ideas for programming and circuit design. But there was also talk about the rebirth of ideas from the sixties, in which, instead of food cooperatives, there would be “knowledge cooperatives”; instead of encounter groups, computer networks; and instead of relying on friends and neighbors to know what was happening, there would be “community memories” and electronic bulletin boards. Computers, long a symbol of depersonalization, were recast as “tools for conviviality” and “dream machines.” Computers, long a symbol of the power of the “big”—big corporations, big institutions, big money—began to acquire an image as instruments for decentralization, community, and personal autonomy.

Personal computers have entered the American consciousness as a new variation on the Horatio Alger story—the story of Steve Jobs and Steve Wozniak beginning with a machine in a garage and turning it into the Apple computer corporation. But in the late 1970s another mythology was also born, as resonant of the 1960s as Horatio Alger was of an earlier period: build a cottage industry that will allow you to work out of your home, to have more personal autonomy, more time for family and the out-of-doors.

Hannah worked as a programmer in a large corporation for ten years before starting her own software consulting company specializing in educational programs for personal computers. To her, nothing is more depressing than working on a tiny piece of a problem. In her old job, “most of the time I didn’t even know what the problem was. With my computer at home I do everything. I get to see all my kinds of thinking. Most important, I have more control over my time, I can spend more of it with my family.”

*CoEvolution Quarterly, Mother Jones, Runner’s World,* and *Byte* magazines lie together on Hannah’s coffee table. The tableau was not unusual. In the
years after they first appeared, people tied personal computers into aspirations for ecology: decentralized technology would mean less waste because people would attend electronic meetings and conferences, do more work out of their homes, and save on transportation and energy costs. Personal computers fueled aspirations for alternative education: schools would become obsolete as computers brought an individualized curriculum into the home. Personal computers became symbols of hope for a new populism in which citizens would band together to run information resources and local government.7

When in the 1960s social critic Ivan Illich searched for an image of how people could have direct access to each other—for learning, for teaching, for bypassing traditional hierarchies—he came up with a giant telephone directory and switchboard. A decade later, personal computers linked on networks provided new technical images to support such aspirations.

There was talk of harnessing computer power for networks and “knowledge co-ops.” Some tried to make it happen. Most did not. The more usual way computers became associated with anti-establishment politics derived from a vague sense that the computer in the basement, living room, or kitchen was a window onto a future where relationships with all technology would be more immediate, where people would understand how things work, and where dependence on big government, big corporations, and big machines would end. But this vague sense did not come only from the idea that the small, personally owned computer could be of practical help in creating new political networks and decentralized information resources. Something more subjective was at work. The first-generation personal computer culture evolved a particular style of working with the machine and the style itself became a political metaphor. People imagined a computer-rich future by generalizing from their particular relationships with home computers. They generalized from a style which, as we shall see, was characterized by transparency, simplicity, and a sense of control.

For some people the railroad signifies progress; for others it signifies the rape of nature. But either way, if you want to use the railroad, you have only one choice. You buy a ticket, get on the train, and let it take you to your destination. This is not true of every technology, and with the personal computer of the mid-1980s it is not the case at all.8 Deborah’s carefully planned-out stars and rectangles were very different from Bruce’s whirling, out-of-control spirals. What was different was not just the product, but the way in which the actual experience of using the machine offered a way to think about who one is and who one would like to be.
The first-generation personal computer owners also used the computer experience to think about issues beyond the self. It was used to think about society, politics, and education. A particular experience of the machine—only one of the experiences that the machine offered—became a building block for a culture whose values centered around clarity, transparency, and involvement with the whole. Images of computational transparency were used to suggest political worlds where relations of power would not be veiled, where people might control their destinies, where work would facilitate a rich intellectual life. Relationships with a computer became a depository of longings for a better, simpler, and more coherent life. Understandings in the realm of personal computers were contrasted with more diffuse understandings elsewhere.

Wayne is around fifty and has always worked as a salesman in electronics. In 1981, he bought his computer on an impulse, but it soon became a preoccupation. Wayne taught himself to program, mobilized friends to teach him about hardware, became a devoted member of his local computer society and a smaller users’ group. Within the space of a few months, Wayne had joined a subculture: he had learned a new language, felt himself to be the holder of esoteric knowledge, and several nights a week were taken up with sharing its collective rituals. What Wayne wants out of his computer is a perfect illustration of what I mean by computational “transparency”: Wayne wants to know “exactly how things work” and expresses sharp frustration at gaps in his ability to follow the system through. “I can’t really follow the continuum. . . . There is a big gap in my own mind from the idea that an electrical circuit can be on or off to the binary number system, and again from there on to the BASIC language. I’ve got to understand all of that.”

Wayne’s desire to understand his system from the and/or gates through the flipflops, the machine language, the assembler and up, carried the emotional intensity of long-standing political frustration.

People used to understand more about how things work. We live in a world where we don’t understand how anything works. I live in an economy, and I don’t understand how things happen in it. I watch the energy crisis, I don’t understand why things are happening that way. . . . I don’t want that to happen with the computer.

In the computer, Wayne saw a chance to develop a depth of understanding that eluded him elsewhere. And then he took his hoped-for relationship with the machine as a standard for other things. Just as the computer is available to understanding, so should be the political world: “Politics is a system, complex to be sure, but a system all the same. If people
understand something as complicated as a computer, they will demand greater understanding of other things.”

I heard many echoes of Wayne’s sentiment. But this tone of “pleasing populism” in the early personal computer culture was problematic. The satisfactions that the computer offers are essentially private. People will not change unresponsive government or intellectually deadening work through an involvement, however satisfying, with a computer in the den. They will not change the world of human relations by retreating into the world of things. There is a tension here. It would certainly be inappropriate to rejoice at the “holistic” relationships that personal computers offer if it turns out that they serve as a kind of opiate. One thing is certain: for the technical hobbyists of that first generation, part of what made the personal computer satisfying was that it felt like a compensation for dissatisfaction in the world of politics and the world of work.

Understanding how the computer can be used in this way requires that we step back and look once again at styles of programming. The “hard masters” among children demanded control. But here, for this community of adults, something new is added to the desire for control. Intellectual fragmentation at work and the complexity and smoke screens of political life create new pressures, and with them a desire to find transparent understanding somewhere. In contrasting hard and soft mastery, the issue was planning versus the pleasures of negotiation. Here another contrast is needed: risk versus reassurance.  

Risk and Reassurance

At the Indianapolis racetrack, cars zoom by in the hands of master drivers highly tuned and sensitized to the machines’ response. Yet many of these drivers end up in fatal crashes. By contrast, there is another kind of automotive mastery, where pride comes from detailed knowledge of cars to help insure family safety. There is mastery in the service of the desire to operate just on the edge of disaster, and there is mastery to feel safe. Cars can kill. Computers don’t. But here too there are safe programmers and “racing-car” programmers, those who pursue risk and those who avoid it.

Risk and reassurance can be played out in programming because the computer presents two possibilities to the programmer: there is the “local simplicity” of the line-by-line program and there is the global complexity of the dynamic processes that can emerge when the program is run. Locally, each step in a program is easy to understand; its effects are well defined. But the evolution of the global pattern is not always graspable. As
soon as a program reaches a certain level of logical complexity, its behav-
ior is no longer predictable by its programmer in any simple sense. One is
dealing with a system that surprises. Depending on how the programmer
brings local simplicity and global complexity into focus, he or she will have
a view of the machine as completely understandable and under control or
as mysterious and unpredictable, even fraught with risk. By concentrating
on the local, the line-by-line, you feel in control. By concentrating on the
global, you see control slip away and can then feel the exhilaration of
bringing it back.

In Carl, the programmer who feels like “just a cog” on the job, we have
met someone who demands an understanding of the whole and a sense
of total control when he works with his computer at home. By contrast,
the “racing-car” programmers enjoy the sensation of playing with risk: the
system might “crash” or behave in unexpected ways; some new relation-
ship between parts of a program might trigger an unanticipated effect in
the hardware of the machine.

We see the spirit of risky programming in Howard, now a university pro-
fessor, who thinks of himself as a hacker. He uses the term hacker to
identify himself with a subculture of programming virtuosos devoted to
programming as an art in itself.

Howard described a fantasy in which he would walk up to any program
and “fix it, bend it to my will.” As he talked, he used the kind of hand ges-
tures a stage magician makes toward the hat before he pulls out the rabbit.
For Howard, what was most thrilling about the experience of programming
was “walking down a narrow line,” using his ability to make small changes
to keep the whole under control while at the same time producing dra-
matic effects. He was always searching for ways to make a local fix, what
he called “neat hacks”—changing some very small thing in a program that
would have a powerful “and in the best case utterly counterintuitive” result
for the system as a whole. His was a magician’s fantasy because what he
was looking for ideally was doing something small, like touching one key
or typing one character, and having the whole system come alive.

Howard’s fevered love of programming, which he describes as the feeling
of “walking near the edge of a cliff,” contrasts with Carl’s approach, which
is geared toward maintaining a safe place. Carl works on well-defined pro-
jects he chooses himself, projects whose beginning, middle, and end are
all under his control. Carl is not interested in mystery, in magic, in local
fixes; when he programs he likes the sure and the explainable, he likes to
see the whole problem in all its detail. Howard works on large, “almost
out-of-control” programs; Carl chooses well-defined, delimited ones. He
describes his home projects as compensation for the alienation he feels on the job and says that he works most intensively on his home projects when he feels furthest away from “seeing how the whole thing fits together at work.” Howard finds documentation a burdensome and unwelcome constraint; Carl enjoys it, he likes to have a clear, unambiguous record of what he has done. Indeed, much of his sense of power over the program derives from his feeling of mastery over its precise specifications. Howard and Carl may work at the same computer, but they approach it with different aesthetics and are looking for different satisfactions.

This is illustrated by the computer language each prefers, the mode each uses to communicate his ideas and intentions to the machine. The computer can be thought of as a set of locations, each designated by a number. These locations are like different “addresses” within the machine. Each address contains a sequence of elements called bits. Each bit can be on or off. If the two states, on and off, are given the names of 0 and 1, then the sequence of elements in any location can be thought of as a number expressed with the digits 0 and 1.

Some simple operations are built into the computer. One is the ability to transfer information stored at the various addresses—for example, to move the contents of one address to another or to move the sum of the contents of two addresses to a third. Another kind of primitive operation involves taking the contents of a given location and treating them as instructions to be acted upon—that is, treating information within the machine as a program. These primitive operations built into the hardware are the machine’s language. Machine-language programming consists of getting the computer to achieve increasingly complicated effects by stringing together the primitive operations. When you use only these to program you are in direct contact with the “bare machine.” This does not mean that the contact is physical. You program in machine language by typing at a keyboard. The closeness of the contact is symbolic—you are talking the only language that the machine can “understand” directly. All instructions need to be designated with 0s and 1s; locations in the machine must be referenced by number. All other “higher-level” languages you might use will ultimately have to be translated back into these long strings of 0s and 1s. But when you use a high-level language you are not involved in that process of translation. It is done for you by a translation device that takes programs written in the high-level BASIC, or Logo, or PASCAL, or FORTRAN and puts them into machine language.

Howard will program only in high-level languages. They let him play with the computer without having to worry about the details of the hard-
ware. Carl knows four high-level languages, and his home system allows him to use two of them, BASIC and PASCAL, but when he works at home he prefers to write in “assembly language,” which is a way of programming one very small step up from machine language. In assembly-language programming, the programmer still has to think in terms of the machine’s primitive operations, but is able to refer to them and to locations within the machine by using easier-to-remember names. For example, if the programmer is trying to add several numbers, he or she may refer to the location in which the sum will appear as “TOTAL” or “SUM” instead of referring to it by a number, like 0000000000100011. In common parlance, when programmers talk about “machine-language programming” they are usually talking about assembly language. With assembly language, you are still very close to the bare machine—only a few mnemonics away. Like a high-level language, an assembly language needs to be translated back to machine language. This is done using a translation program called an “assembler.” People like Carl who enjoy contact with the bare machine and its logic often increase their contact by writing these translation programs themselves.

Carl does not justify his preference for assembly language in instrumental terms. He does not speak of how fast his assembly-language programs can run. His reasons are subjective: working in assembly language gives him the feeling of having direct contact with what is “really going on” in his computer. The safety he seeks is a symbolic safety, more like the kind that Deborah worked toward when she constructed her “thirty-degrees world.” Carl wants to feel in close contact with machine logic; he wants the reassurance of step-by-step mastery. Howard thinks that this kind of contact is a “total waste of time.” He doesn’t even think of it as mastery of the computer. It is simply tagging along after its machine limitations. For Howard, programming is the enterprise of transcending these limitations. Programming as magic means programming as transformation.

In the high-level languages Howard prefers, the programmer no longer has to think in terms of locations or of machine-level operations, and there is no longer a transparent relationship between the steps written by the programmer and events taking place in the machine. The logic of high-level languages is adapted to how people think, not how the machine “thinks.” For example, in the high-level language BASIC, an instruction might read “LET Y = SIN X.” In Logo, we saw the instructions “FORWARD 100” and “RIGHT 90” and “SETCOLOR :RED.” These instructions are far from the 0s and 1s of machine locations. Just as an assembler stands between an
assembly-language program and the machine language, a much more complex translator (some are called compilers, some are called interpreters) stands between a high-level-language program and the machine language. When you program in Logo or BASIC, the internal structure of the machine need never cross your mind. The compiler or interpreter takes care of it for you. High-level languages take the programmer away from contact with the bare machine. For Carl, this feels like a loss; for others, like Howard, a liberation. The differences are not simply of individual taste. Preferences in programming language and programming style are building blocks in the construction of computer cultures, in this case the culture of the first-generation hobbyist and the hacker.*

One might see the differences among computer cultures as expressions of differences in the typical styles of their members’ personalities. For example, a young hacker might spend time with other hackers because of shared personality traits such as a need to assert control that is so strong, so absolute, that it can be expressed only in relation to things. But once in their company, he or she becomes integrated into a community that amplifies whatever shared features of personality brought its members together. What might have begun as an expression of individual personality develops into a social reality.

Carl sought safety by making the computer a protected, intelligible world. In this he seeks what many first-generation hobbyists I met looked for in their machines. Sometimes they found it in assembly-language programming, which they described with such phrases as “I like the feeling of knowing I can optimize better than any dumb compiler” and “It makes me feel calm. Having control from the bottom level of the program and up makes me feel comfortable, safe, sort of at home.” Sometimes the hobbyists found their sense of safety in relationships with the computer’s

* High-level languages have another quality that makes them attractive in one way, yet less satisfying if you come to them with Carl's desire for transparency. Each of them is designed for a special kind of programming. For example, BASIC is designed for algebraic manipulations, COBOL is designed for manipulations such as sorting records in business applications. Programs of the kind for which the high-level language is specifically designed can be written more quickly and economically in it. But other kinds of programs might be cumbersome or even impossible. A car is a superior means of transportation if you know that you are going to be traveling paved roads. If you will be traveling up a mountain trail or across sand dunes, you might do better to have a horse. The computational analogy to the horse as a "general-purpose" aid to transportation is assembly language.
hardware. Carl himself is deeply involved in a plan for revamping his home system’s hardware. He built his system from “components”—a processor, a monitor, a keyboard, a printer. Such a component system needs to be connected with interface devices, usually a simple circuit. In Carl’s case, each interface circuit needed only a few specialized and inexpensive chips. But Carl had a plan to use a separate microprocessor as an interface for each component. There was no need for a general-purpose microprocessor at each interface; it was “overkill” in terms of computer power. But the specialized chips that could do the job for less money could not satisfy his desire to build a transparent, maximally coherent system. To him, the special-purpose circuits were “black boxes.” You could not see inside of them. By replacing the black boxes with separate, identical general-purpose microprocessors whose workings he understood, the system, at least in Carl’s eyes, became uniform, intelligible, and elegant. Carl had conflicts about his project. He felt uncomfortable because he could not justify it “rationally.” But from the perspective of Carl’s subjective relationship with the technology, his rationale was clear. The “inexpensive” solution, a collection of opaque, ad hoc circuits, felt unintelligible. The inexpensive solution reminded him of the kind of relationship with computers he had at work. His plans at home were dictated by the logic of psychological compensation rather than by the logic of material economy. Most important was working in an environment where there were no black boxes and “things didn’t change unless you wanted them to.”

Of course, sometimes things did “change.” The first generation of personal computers had problems. There were false starts and frustrations. There were bugs in the machines. But even if they couldn’t be found, bugs became “known,” “familiar,” even reassuring. Joe owns a secondhand computer “with a lot of hardware problems.” He likes to envision the relation of his program to specific actions taking place within the core of the computer, its central processing unit, its CPU. When Joe suspects a bug, he pulls out an oscilloscope and checks whether the CPU is doing what it should in response to a given instruction. He figures out what the signals should be doing and collects evidence for what is going wrong. He traps the bugs himself. Sometimes he fixes them. Mostly, he “just finds out what is going on.” He says the bugs in his system “have become almost like friends. I turn on the machine and I check out my ‘old friends,’ and I swear, finding them there has a certain reassuring element.”

What is reassuring here goes beyond predictability. Joe, like Carl, is looking for a direct relationship with the CPU, the “body” of the machine.
If the “mind” of the computer is that part of computation which involves thinking in terms of high-level programs, then relating to the body of the computer means not only working on hardware, but also working with programs in a way that is as close as possible to the machine code, the language the “body” understands.

Not everyone is drawn to or could even tolerate the fastidious, meticulous work and record-keeping that so reassure Carl, Wayne, and Joe. The desire for this kind of understanding, the desire for an as-close-as-possible relation with the CPU, reflects individual personality. But it reflects other things as well. For many pioneers of the personal computer culture, this style of relating to the computer was “overdetermined” in the sense that a host of other, more general forces also came together and were expressed through it.

To begin with, it had an intellectual dimension. The first-generation personal computer owners were for the most part men who had always been interested in technical things and had a long history of involvement with computers. It has become a cliché to say of microcomputers that they follow the developments of larger computers with a ten-year delay. These people knew what was state-of-the-art and where the art was going. For them, the early personal computers seemed incredible, even awesome, because of their low cost and small size, but there were many things about them that seemed second-best—a collection of hand-me-down pieces, a collection of patches (in computer jargon the word is “kluges”) dictated by arbitrary corporate decisions, by economic necessity, by the manufacturer’s need to cover over past mistakes. The corner of the personal computer which they felt had the greatest intellectual integrity, which distilled the best ideas in computer science, was the CPU.

Whether or not they ever pull out an oscilloscope as Joe did and follow their machine-language programs step by step as its instructions pass through the CPU, getting close to the bare machine makes them feel in touch with what is most “pure” in the computer. They built a computer culture around a widely shared aesthetic of simplicity, intelligibility, control, and transparency. It is not surprising that for them, involvement with the “unspoiled” part of the machine holds the greatest satisfactions.

A second dimension is more directly psychological, related to the deep roots of computer “holding power.” When children program the Logo turtle to make designs on the screen, or if they are using the small robot that has come to be known as a “floor turtle” to trace designs on paper on the floor, they are told to “think of themselves as the turtle.” They are asked to “play turtle” and to make their design happen by instructing the
turtle to do what they have to do in order to trace the same pattern with their bodies in space.

When you watch children learning to program in Logo, you watch children moving, tracing shapes, reflecting on what they have just done and trying to make the turtle make the same moves. The turtle has holding power because there is what Seymour Papert has called a “body syntonic” relationship between it and the programmer. When people become involved with the CPU, here too the relationship can be physical, although this is not as immediately apparent as when we look at children and turtles. Here too there is a body-to-body connection. The CPU’s primary activity is moving something that is conceptually almost a physical object (a byte of information) in and out of something (a register) that is conceptually almost a physical place. The metaphor is spatial, concrete. One can imagine finding the bytes, examining them, doing something very simple to them, and passing them on. This kind of identification is a powerful source of computer holding power. People are able to identify physically with what is happening inside the machine. It makes the machine feel like a part of oneself. It encourages appropriation of the machine as a tool in Marx’s sense—as an extension of the user.

Third, this relationship with the CPU as extension of self is all the more powerful because it is in contrast with the “other computer” that people know at work. There they relate not to the tool, but to the machine. I have noted that many members of the first generation of personal computer owners worked in the middle echelons of the computer industry. A bureaucracy stands between them and the computer; a bureaucracy schedules the computer, decides its up and down time, apportions the work for its software design, and decides on priorities and procedures for access to it. At work, when something goes wrong with the system it is usually the fault of an intermediary person, yet another “someone else” who deals with the machine, or it may be the fault of an intermediary in the machine itself, a compiler, an interpreter, an operating system; all of these are someone else’s program. At home, what is savored is the opportunity to work directly with the CPU. When something does go wrong, it is between oneself and the bare machine.

When the early owners of home computers bought their machines, their software and hardware were in a state of disequilibrium. The CPU was elegant and transparent but the high-level languages available on the first small machines were opaque, arbitrary, and primitive. The interpreters and compilers that allowed them to run were designed to optimize what you could do on a small computer. They were designed to be practical, not to
be understood. Everything was in place, then, for the first generation of personal computer owners to develop a relationship with the bare machine and to see it as a model for transparency and comprehensibility. Everything was in place for the technology to attract a group of people who had a particularly strong need to work through frustrated desires for that kind of transparent understanding. And everything was in place for them to use their relationships with personal computers as a metaphor for a new grassroots politics.\textsuperscript{10}

\textbf{The Next Generation}

Technical hobbyists dominated the early personal computer culture: they gave that early culture a certain coherency and supported one of its dominant themes, the computer as a way to counteract feelings of political alienation and discontent in the workplace. The people who bought their first personal computer in the 1980s purchased a different machine than was available to the first generation of owners. They are not coming to the computer during a time of heightened political awareness. And for the most part, they do not have technical backgrounds.

When the personal computers first appeared, nontechnical people were intimidated and kept their distance. In their worlds, “computers” continued to be the electronic teller at the bank, the data-retrieval system at the office, the word processor on the secretary’s desk. But increasingly, the computer became less compartmentalized. Millions of Americans brought them into their homes, encouraged by the increasing availability, reliability, and portability of the machines, and encouraged, too, by their decreasing cost.\textsuperscript{11}

The interest of this second generation of personal computer owners is being elicited by advertising that associates computers with new and appealing images. For example, from the beginning, Apple had an image of being a counterculture computer. Its name and its logo suggest nature and simplicity. When IBM came into the field, it sensed the need for a strong symbol to break away from its Brooks Brothers image. Its promotions made an icon of antitechnological innocence into the trademark of its personal computer: the Charlie Chaplin tramp, Charlot with a rose.

Beyond being sold as an “antitechnology technology,” personal computers are being presented as a technology for the young, the chic, the successful, who build their own companies and make their own schedules.\textsuperscript{12}

Onto such complex images manufacturers project more concrete promises as well: the machine will help Father with his finances, Mother with
her writing, the children with their schoolwork. The machine is presented as a way of asserting status, a way of saying that this is someone who has not been left behind. Few purchases carry so much expectation. Even before the purchase is made, people start spending time “on the computer,” because the decision to buy one carries the question “Which one?” It is unusual for people to come to a major purchase with so little prior knowledge. It is unusual for people to come to a major purchase with so little confidence in their own taste.

Just contemplating owning a personal computer means entering a new world of information to be gathered and assimilated and discussed. For most people, the portal to that new world is the personal computer store. Considering the purchase already means that one finds oneself in debates about 16K versus 64K, about disks versus tapes, about high resolution versus character graphics, about dot-matrix versus daisy-wheel printers.13

There is something striking about these conversations. People seem to feel a pressure to have an opinion about all such matters. There is a reluctance to treat them as technical decisions to be left to a technical expert. Becoming fluent in this language, participating in this world, is part of what people are buying.

In the case of the technical hobbyists, the computer rarely served a truly instrumental function—the point of having the machine was the pleasure of working with it. For nontechnical users—and the typical home computer owner of the 1980s fits this description—the machines are used for a purpose: games, word processing, record keeping, reinforcing school learning. But here too, once they are in the home, personal computers get taken up in ways that signal the development of something beyond the practical and utilitarian. People buy an “instrumental computer,” but they come to live with an intimate machine.

Carl built interface cards and programmed his computer in assembly language. The concerns of today’s computer owner who uses the machine to play games or as a word processor appear to be far from his. And yet the language people use for talking about their personal computers, and the feelings the machines evoke, recall the first-generation hobbyists. In some cases the similarity comes through loud and clear, in others one has to listen carefully for it. But in either case, I have come to see Carl and his peers as more than members of a somewhat exotic and isolated subculture. They offer a window onto an aspect of how many of today’s home computer owners relate to their machines.

David, the lawyer who found Zen in the video game, talks about buying a home computer to play games. What he says about their satisfactions is
reminiscent of Carl’s feelings about assembly language and Wayne’s desire to “follow a whole system through.” Like many of the first-generation owners, part of what the computer means to David is a chance to recapture the sixties. He complains about the conventionality of his life. He calls it “music you could play at anybody’s wedding.”

Like Wayne, David does not feel he is in a relationship with the world where he understands how things really work: “People don’t build cars, people build transmissions or people build widgets that go inside the transmissions of cars. No one ever gets to see the whole thing that they’ve done.” Like Carl, he is frustrated with not dealing with the whole on his job:

After I’m done taking care of your divorce or some guy’s criminal problem, that’s the end of my involvement with that person. I can’t have the follow-through to find out the end of the story, which could be that he went out and got arrested in Nevada and somebody else is representing him, or he lived happily ever after. I get fragments, always fragments, of people’s lives.

David likes the games because they form a transparent, completely knowable world. “It is a small one, maybe not a very important world, but you get to know it all the way”:

You win something for yourself. You lose something for yourself. You live out an existence. Even if it’s just the existence of the character you are playing, you get to live it out in its entirety. You get to see the whole picture. There’s nothing more to the game than what is there. There’s nothing that you don’t have control over.

David complains about not being closer to nature and dreams of moving toward “something more real.” The irony, of course, is that for David a simulation, a computer game, is what comes to feel most real.

A forty-five-year-old editor named Gerald has similar feelings, in his case not about games but about using his new personal computer for word processing. Like David, Gerald comes from a nontechnical background and has always seen himself on the other side of a divide that separates him from people who are good at sciences, good at math, and “good with machines.” He tells me that the computer has changed his life. He talks about an experience that first-generation owners called “coming to see all of your kinds of thinking.”

Gerald quickly went beyond using ready-made word-processing programs. He learned to customize his system to suit his own needs. And as he modified his system, he modified his sense of not being able to work in domains that require precision and clear-cut decisions.
As a kid in school, I was always interested in literature and history, the “soft subjects.” But as soon as I hit high school I started to get really jealous of the science types. Not because I wanted to do it, but because you could be so sure of yourself. You had a right answer. You could go for the right answer, and if you got it, no discussion. I was so insecure. I always felt that that kind of certainty, the science certainty, would be great for me. But I thought it was beyond me. At my job as social science and philosophy editor, there is always ambiguity. The art is in the judgment call.

I got a computer when they began to seem reliable, when you wouldn’t be spending all of your time hunting after some genius to fix it, and I started to do programming right away. For me it’s the first time that I have that feeling of knowing the right answer, of understanding everything that’s happening. It’s a real break from the rest of my “everything is relative” life.

Thus, when people in the mid-1980s speak about the machines they have brought into their homes, their way of talking about them has many similarities with the discourse of the first-generation hobbyists. With David and Gerald we see a first: the computer is looked to as a medium of compensation for what is not found at work. There is a second: although the computers available and the population buying them have changed, there still seems to be the impulse to find a way to a sense of intimate understanding of the logic of the machine.

The first-generation home machines were easy to “open up,” to peer into and experiment with, but the new machines tend to be closed “black boxes.” They are built as a technology to be exploited, not explored. Manufacturers, now in the business of mass marketing to consumers not likely to be experts, are worried about warranties, guarantees, and parts replacement. Hardware comes in boxes marked “Do Not Open,” “Not User Serviceable,” “Warranty Is Void If Seal is Broken by Owner.” And if you do open the box, you find that behind the physical inaccessibility of the hardware there is a new intellectual inaccessibility. The first personal systems were built up in a modular way. You could follow function through form. With mass production, with greater compression of function on single chips, intelligibility is sacrificed to ease and cost of fabrication.

At the same time that the hardware is growing more “opaque,” a greater and more powerful variety of languages, operating systems, and other software is becoming available: text editors, file management programs, and debugging aids. Such systems give increased computing power. But the increased power does not lead to a sense of direct control where nothing stands between the programmer and the bare machine.
Despite all this, many people make active efforts to bypass the new features to get back to a more direct sense of contact.* As we have seen, the “direct-control” style does not necessarily depend on physical contact with the hardware. When you program in machine or assembly language, you are typing at a keyboard. The sense of physical relationship depends on symbolic contact.

Arthur, thirty-four, is an architect. He owns one of the most up-to-date personal computers on the market. He bought it “off the shelf” at a computer store and had it set up to do useful things. It has an elaborate word-processing system that he uses for writing proposals and reports, and it has a special program to do calculations necessary for his design work.

After one month Arthur started to teach himself to program. “I think of myself as a competent man. Programming seemed like something I would probably be good at.” He learned BASIC and PASCAL and then wanted to do more. For him, this didn’t mean learning other high-level languages or starting to work on larger or more complex programming projects or machines. Expansion was to be vertical, not horizontal. “Pretty soon programming in BASIC and PASCAL started to feel elusive. Like I wasn’t in touch with where it was really happening. I wanted to get down there and play with the machine. I liked getting inside, changing things around, seeing that I really understood them.”

Arthur became a master of “peeks” and “pokes.” These are instructions in high-level languages that reach down to the level of the bare machine. Normally, when you work in a high-level language, you do not think in terms of machine events or actual locations in machine memory. But the “PEEK” instruction allows the programmer to look into a specific memory location and see what is happening in it. And the instruction “POKE” allows the programmer to act on a location, by inserting or “poking” a specific byte into it. These are ways of breaking out of the limitations set by the high-level language. Arthur was drawn to the CPU because he wanted to “get inside” and be able to “move things around.” “This kind of programming is a real high, a very sexual thing. Sometimes I feel guilty when I do it for too long.”

When people repair their bicycles, radios, or cars, when they build their own stereos, they are doing more than saving money. They are surrounding themselves with things they have put together, things they have made

* Indeed, part of the success of the Apple II computer seems to have been due to the fact that it was designed to make it easy for people who wanted to get at its “innards.”
transparent to themselves. Bicycles, motorbikes, cars, and radios are one thing, but microprocessors, which cannot be taken apart, whose structures are microscopically etched on silicon chips, are another. Nevertheless, we have seen that many personal computer owners insist on treating their computers like bicycles. They labor to make the computer transparent. Professional computer scientists work to develop technology that does not demand technical knowledge of its user, to develop “human interfaces” that will make relating to computers more like holding a conversation and less like taking apart a bicycle. But, at least for now, many are unwilling to accept the computer as an entity whose underlying structure can safely be ignored. They demand some sense of understanding the machine. Of course, for different people what counts as understanding is different. For Arthur, it was directly linked to what the first-generation hobbyists searched for, the “body” of the machine. For someone like Doris, it is less direct.

Doris is a professor of history and has never felt comfortable around technical things. She was able to get good grades in math courses in school, but says, “I had no sense of really understanding the symbols I learned to push around.” She bought a computer not out of any particular interest in the machine but because of an urgent deadline on a writing project and a friend who argued that she would write faster if she had one.

In Doris’s first tentative probings of her word-processing system, she used a cookbook method, making lists of rules on cards she kept at her side. Little by little she found herself playing with the rules, seeking alternative ways of getting the same result: moving a paragraph, filing key words to use later in an index, and so on. A computer scientist looking at Doris might say that she is not “really” learning about the computer, because she is not programming or learning about the machine’s internal structure. The computer scientist would be wrong. There is another reality: Doris is acquiring a sense of what it is like to work within a formal system.

The art of writing a program (like the one Doris uses for word processing) that must be accessible to the computationally unsophisticated, consists of designing an abstract, artificial world so that it feels like a familiar physical one. When you use a word-processing program you become familiar with an environment in which there are “places” analogous to files, “objects” analogous to sheets of paper, and “operations” analogous to copying, cutting, pasting, and filing.

The analogies with physical reality helped Doris get started. She could pretend that she was in the physical world with small differences: instead of using an in–out tray for temporary storage, she used a “place” called a
buffer; instead of placing her text in a file, she gave an order for it to be placed there. But Doris soon discovered that the analogies with the “real” were not precise: what ruled here were the laws laid down by the programmer. Some simple operations produced results as if one were through the looking glass. If you give two documents the same name, one vanishes. If you display a document on your screen, it stays where it came from as well. Slowly, without any sense of a conscious break into a new way of thinking, Doris learned the ways of a formal system—she began to learn the peculiarities of a purely logical universe, one defined entirely by rules. And her pleasure in doing so came from her belief that this universe is in principle totally understandable. In this, although she does not program, Doris’s relationship with the computer had much in common with what programming means to those who approach it in the spirit of the first-generation hobbyists.

Doris did more than get to know one constructed world. She was doing her writing on a general-purpose computer, rather than the much more restricted special-purpose word-processing machines that are common in business settings. This meant that she also got to work with other programs. She needed to use a separate program in order to prepare the disks on which her text is kept, to make “back-up” copies of her work, to rename files, and to combine them to make her chapters into a book manuscript. Her involvement with these many programs gave her a sense of accomplishment at being able to find her way around not one constructed world, nor, indeed, a number of them, but within a system of logical worlds whose unity is symbolized by the “operating system,” the program that coordinates and gives access to the other programs in the system. At first, using the operating system felt to her like the symbol-pushing of her memories of school mathematics. But as time went on, its rules and hierarchy began to seem elegant, its patterns reassuring. Like Carl, Doris found pleasure in understanding a system in that especially complete way that does not ever happen in the “real” world. Doris’s manipulations of the operating system, like Barry’s manipulations of mathematics in his fingers, like Anne’s painterly manipulations of the “sprites” in the Logo system, offer a tactile, “soft” access to a world of hard rules. The question of how this point of access can be transformed into fluency with other things scientific and mathematical is an open one. But a threshold has been crossed.

There is something else notable about Doris’s relationship with her computer: she brought it into her life to write a book, but it brought her into a culture. Doris attends a users’ group, she communicates regularly with the other faculty at her school who own her brand computer, she sub-
scribes to two personal computer magazines, one that features news about her computer and another more general in scope. She is even beginning to think of herself as a particular kind of computer user, different from others. Although Doris does not program, she is clearly on one side of the divide separating the style of the “prototypical hacker” from that of the “prototypical hobbyist.” She is interested not in magic, but in transparent understanding:

I have a friend whose son wants to be a computer wizard [the term used at her university for virtuoso programmers in the style of hackers]. For a while, I called on him whenever I had a problem. But I can’t stand the way he works with the computer. He won’t read the sections of the manual I show him. He sits down and starts typing. He never seems to know exactly what’s going on, but somehow by instinct he finds a way to solve the problem. I can’t stand it. I have stopped asking him for help.

Doris’s style recalls the first generation of hobbyists, a style that puts a premium on transparent understanding. And she knows enough about what she enjoys in the machine to feel a sense of difference from her friend’s son, the incipient hacker. Most people think of a computer as one thing for all people, and so, when they become aware of this kind of differentiation, it surprises them. Doris’s initial reaction was to treat it as a matter of who knew more about the computer, and then as a manifestation of arrogance on the part of the young wizard. But something more is at work here.

Doris is beginning to get the idea that although she and the wizard use the same computer, she belongs to one culture and he to another.

**Personal Computation and Personal Philosophy**

In this chapter I have “tagged” two different styles of relating to the computer—one that focuses on magic, the other on transparency—by associating them with the culture of computer hackers and first-generation computer hobbyists. But these relational styles have a life of their own. They exist outside of these cultures. The story of Doris and her friend’s son illustrates that now, within the personal computer culture, there are a multiplicity of styles.

These styles enter into programming and into the computer owner’s feelings about what makes the machine consequential, what makes it satisfying and “beautiful.” I discovered that they also find expression in something else as well: how individuals use the computer to think about other questions, among them “metaphysical” ones.
Children find in the question “Are computers alive?” a way to talk about the line between computers and people. Adults don’t. Nonetheless, they are affected by the questions that stand behind questions such as “What is life?” “What makes us special?” “How do computers challenge our definition of ourselves?” One way of getting adults to talk about these things is to ask not about “life” but about the possibility of machine intelligence. And here different styles of relating to the computer correspond to different kinds of answers.

Among the children at the Austen School it is Anne, the most advanced of the “soft” programmers, and Henry, the young hacker who put the highest value on magic, who are ready to ascribe to the computer a “sort of” life. In their own way, each tries to keep the computer mysterious. They do not try to understand it completely. Perhaps they even try not to understand it completely. Like the people who coaxed the ELIZA “psychotherapist” program into making lifelike responses, like Lucy who made Speak and Spell obey her spoken orders, they seem to have a stake in experiencing the machine as autonomous. By contrast, Jeff, a prime example of a “hard” style, found the question of the computer’s life too “dumb” to discuss: computers were obviously not alive. This is not a simple matter of hard masters being more hardheaded. We saw that Jeff’s style of programming has much in common with that of the first-generation hobbyists: it is an element in a coherent system of thinking about the computer as transparent and under control. As such, it shares nothing with the unpredictability of living things.

Adults express their ideas about what computers share with people not through opinions about machine aliveness but about machine intelligence. Committed to a notion of the computer as ultimately comprehensible in terms of its “specs,” most first-generation hobbyists ruled the enterprise of artificial intelligence out of court: “How can you program something that can’t be reduced to specs?”

Today people like Doris are drawn toward a relationship with the computer equally committed to keeping it unmysterious. And this often brings them to that same position: computers are too “mechanical” to have anything in common with mind or with life. What is powerful about the computer is placed in a different realm from the human mind. According to the aesthetic of what we might call the “pure” hobbyist culture, human intelligence has a quality of mystery. It is precisely what cannot be reduced to specs. It is precisely what cannot be meaningfully analogized with computational processes. One computer owner put it this way: “In a computer,
no matter how fancy, all that is going on is GIGO, ‘Garbage in and garbage out.’ You can have all of these programs, you can even have them talking to each other, but in the end, you told them to do it. You can’t have a spark of life in a computer. That spark of life in people, well, that must be God.”

Other computer cultures—for example, the culture of artificial intelligence researchers or the hacker culture to which the “risk-taking” programmer Howard said he had once belonged—have very different computational aesthetics that can lead them in very different directions when they use what they know about computers to think about themselves. Recall that Howard’s sense of real computer power is more like that of Henry and his mysterious inventions—for them it is somehow incompatible with specs. Far from finding artificial intelligence irrelevant, Howard sees it as an embodiment of what he finds most exciting about the computer: the way in which unpredictable and surprising complexity can emerge from local ideas. Howard believes that if you make the system complicated enough, the simultaneous operation of millions of interacting programs might be able to create in a machine “that sense of surprising oneself . . . that makes most people feel that they have free will, perhaps even . . . a soul.” Someone like Carl sees computers as mechanical. Either people are the opposite or, if they are to be analogized with computers, the human psychology that emerges is resolutely mechanistic. People like Howard see the machine as complex and unpredictable. They want to see the machine as like the human and want to see the human as like the machine.

In the early days of personal computation, the world of the hobbyist came close to being a distinct and homogeneous computer culture. But as personal computers enter the lives of wider groups of people, this culture has been overwhelmed by centrifugal forces: an increasing diversity in who participates and why—and “participation” can come to mean no more collective activity than attendance at an occasional meeting. By contrast, the following chapter describes a group that has developed in conditions favorable to its growth as a culture: it shares a unity of place, lifestyle, and passion. It has developed its own rituals, language, myths, even its own literature. This is the culture of virtuoso programmers known as “hackers.” Hackers are often described in the media as people whose involvement with computers has drawn them away from involvement with other people; in fact they are drawn away from people who don’t belong to their world, but within it they form a tight web of relationships where the computer is the center of an all-embracing way of life.