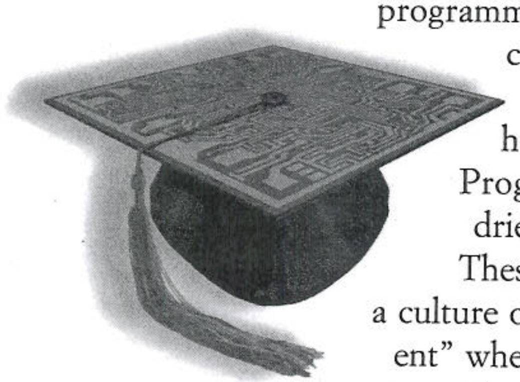
SEEING THROUGH COMPUTERS

EDUCATION IN A CULTURE OF SIMULATION

BY SHERRY TURKLE

Today nearly everyone is certain that schools and universities should teach students about computers, but exactly what they should teach isn't so clear. The ideal of computer literacy, of an empowering relationship with the computer, has changed dramatically since educators and their critics first began worrying about making Americans computer literate two decades ago. Originally, the goal was teaching students how computers worked and how to write programs; if students could understand what was going on "inside" the computer, they would have mastery over it. Now the goal is to teach students how to use computer applications, on the premise that if they can work with the computer, they can forget what's inside and still be masters of the technology. But is that enough? And might it be too much in some fields of education where using computers is almost too easy a substitute for hands-on learning?

The uncertainty about what students (and the rest of us) need to know reflects a more general cultural change in the understanding of computers. When I first studied programming at Harvard in 1978, the professor introduced the computer to the class by calling it a giant calculator. No matter how complicated a computer might seem, what happened inside it could be mechanically unpacked. Programming, the professor reassured us, was a cut-and-dried technical activity whose rules were crystal clear.



These reassurances captured the essence of the computer in a culture of calculation. Computers were thought to be "transparent" when the users could look beyond the magic to the mechanism. The first personal computers of the 1970s and early 1980s, like the mainframes and minicomputers, required users to know how to issue exact instructions. Someone who knew programming could handle the challenge more easily. By the mid-1980s, increased processing power made it possible to build graphical user interfaces, commonly known by the acronym GUI, which hid the bare machine

For subscriptions and bulk reprints, call 1-888-MUST-READ.

from its user. The new opaque interfaces—the first popular one on the mass market was the 1984 Macintosh—represented more than a technical change. The Macintosh "desktop" introduced a way of thinking about the computer that put a premium on the manipulation of a surface simulation. The desktop's interactive objects, its dialogue boxes in which the computer "spoke" to its user, pointed toward new kinds of experience in which people did not so much command machines as enter into conversations with them. In personal relationships, we often interact without understanding what is going on within the other person; similarly, when we take things at (inter)face value in the culture of simulation, if a system performs for us, it has all the reality it needs.

In 1980, most computer users who spoke of transparency were referring to a transparency analogous to that of traditional machines, an ability to "open the hood" and poke around. But when users of the Macintosh talked about its transparency, they were talking about seeing their documents and programs represented by attractive and easy-to-interpret icons. They were referring to an ability to make things work without needing to go below the screen surface. Today, the word "transparency" has taken on its Macintosh meaning in both computer talk and colloquial language. In a culture of simulation, when people say that something is transparent, they mean that they can see how to make it work, not that they know how it works.

Most people over 30 years old (and even many younger ones) have had an introduction to computers similar to the one I received in my first programming course. But children growing up with computers today are dealing with objects that suggest that the fundamental lessons of computing that I was taught are wrong. The lessons of computing today have little to do with calculation and rules; instead they concern simulation, navigation, and interaction. The very image of the computer as a giant calculator has become quaint. Of course, there is still "calculation" going on within the computer, but it is no longer widely considered to be the important or interesting level to focus on. But then, what is the interesting and important level?

This article continues the series "The New Media and Learning," which opened with three articles in our July-August issue. All articles were originally presented at a conference sponsored by *The American Prospect* at the MIT Media Laboratory on June 4, 1996. Audiotapes of the presentations and discussion are available by calling 1-800-872-0162 or by ordering from our Web site (http://epn.org/prospect.html). The conference and articles were underwritten by a grant from the Spencer Foundation.

What's in an Algorithm?

Through the mid-1980s, when educators wanted to make the mechanism transparent, they taught about the logical processes of the computer's inner workings, typically beginning with an introduction to binary numbers, and instructed children in programming languages that would make computational processes transparent to them. In the highly influential *Mindstorms: Children, Computers, and Powerful Ideas*, published in 1980, Seymour Papert of the Massachusetts Institute of Technology wrote that learning about the computer should mean learning about the powerful ideas that the computer carries. In the Logo programming language he developed, children were taught to give explicit commands to a screen cursor known as a Turtle: FORWARD 100; RIGHT TURN 90. The commands cause the Turtle to trace geometric patterns that could be defined as programs. The idea behind the exercise went beyond the actual programs; Papert hoped that the process of writing these programs would teach children how to "think like a computer." The goal of the exercise was to experience procedural thinking and to understand how simple programs could be used as building blocks for more complex ones.

Although Logo is still in use, educators now most often think of computer literacy as the ability to use the computer as an information appliance for such purposes as running simulations, accessing CD-ROMs, and navigating the Internet. There is certainly nothing wrong and much that is right with students having those skills. But many teachers question whether mastery of those skills should be the goal of "computer education" or "computer literacy."

"It's not my job to instruct children in the use of an appliance and then to leave it at that," says an unhappy seventh-grade teacher at a June 1996 meeting of the Massachusetts chapter of an organization of "Computer Using Educators," a group known as MassCUE. Most of the 80 or so teachers present have been in computer education for over a decade. In the 1980s, many of them saw their primary job as teaching the Logo programming language because they believed that it communicated important thinking skills. One teacher describes those days: "Logo was not about relating to the hardware of the computer, so it wasn't about how the computer 'worked' in any literal sense, but its claim was that it could teach about procedural thinking. It could teach about transparency at its level."

Another adds, reflecting on Logo: "The point was not that children needed to understand things about the simplest level of how the hardware worked, but that things needed to be translated down to an appropriate level, I mean, a relevant level." Someone asks how she knows what is relevant. She stumbles, and looks around to her fellow teachers questioningly. A colleague tries to offer some help: "You have to offer children some model of how a computer works because the computer needs to be demystified. Children need to know that it is a mechanism, a mechanism that they control."

By now, the conversation is animated. Several teachers disagree, arguing that teaching that the computer is a controllable mechanism is not enough. One says: "Children know that the telephone is a mechanism and that they control it. But it's not enough to have that kind of understanding about the computer. You have to know how a simulation works. You have to know what an algorithm is." The problem, however, may be that a new generation no longer believes they have to know what an algorithm is.

Walking Through the Machine

The changing exhibits at Boston's Computer Museum illustrate the evolution of ideas about how to present computers and the dilemmas that educators now face. Oliver Strimpel, the museum's current director, proposed the idea for a "Walk-Through Computer" exhibit in 1987 when he was director of exhibits. Strimpel describes his original idea in the language of a computer transparent to its users: "I wanted to blow up the computer so that its invisible processes could be made visible. I wanted people to understand the computer from the bottom up." The exhibit opened in 1990, its trademark a room-size computer keyboard, a keyboard kids could play on.

At that time, the exhibit began by introducing the visitor to a computer program that charted the shortest route between two cities, *World Traveller*. All that followed was designed to help the visitor trace how a keyboard command to *World Traveller* was translated to lower and lower levels in the machine—all the way down to the changing patterns of electrons on a computer chip. "The key to my thinking," says Strimpel, "was the idea of levels, of layers. We worked very hard to show several levels of how a computer worked, trying to take visitors along the long conceptual path from the behavior of a program to the anatomy of the hardware and low-level software that made it all work. We built 'viewports' that attempted to give people a look inside key components such as the CPU, disk, and RAM."

By 1995, it was time to update the exhibit. The museum's studies of visitor reaction to the original exhibit had shown that many people went through the exhibit without understanding the notion of layering or the message of the viewports. In focus groups conducted by the staff, children said they wanted to know what "happened" when you touched a key on a computer. Their question encouraged Strimpel to go into the first planning meetings committed to a new exhibit that would show the translation of a keyboard stroke into a meaningful signal—the connection between the user's action and the computer's response. He imagined that with improved technology and more exhibit experience, a new version of the walk-through computer could communicate layering in a more sophisticated way.

But Strimpel, in his forties, a member of the "culture of calculation," did not prevail. The people on his staff, mostly in their twenties, were products of the culture of simulation. "What seemed important to them when we went to our second version," says Strimpel wistfully, "was explaining the functionalities—what a disk drive does, what a CD-ROM player does, not how the chip worked. The revised exhibit does not attempt to give explanations at different levels." In the culture of simulation one does not dwell on how the computer solves "its" problems. What is important is that it solves your problems. Strimpel had insisted that the original walk-through computer stress the notion of algorithm. "You could look into a blow-up of how information was passed from one part of the program to another as it attacked the problem of finding the shortest distance between two points," says Strimpel. "In the second exhibit, the idea of algorithm dropped out."

In the revised exhibit, the presentation of a giant, walk-through machine was maintained, updated now to look more like a modern desktop

PC. The walk-through computer had quickly become the museum's trademark. But its function was now purely iconic. As Strimpel puts it, "The giant keyboard became a piece of sculpture."

Boston-area schoolteachers regularly take their students to the Computer Museum. They praise the richness of its special exhibits, the many chances it offers for students to try out computer applications to which they would not otherwise have access. Students learn how buildings and cars and turnpikes are designed. They play with voice recognition and artificial intelligence. Teachers praise the museum's internet exhibits; their students can go online at speeds and with display technology that they cannot even demonstrate in their schools.

But at the MassCUE meeting, the very mention of the walk-through computer provokes heated debate. Several teachers remark that children get excited by the exhibit, but other teachers are skeptical. One comments: "Sometimes, the fifth graders go through that and ask, 'What were we supposed to learn?' But what's worse is that lots of them don't even ask what they were supposed to learn. They're used to the computer as a black box, something you take 'as-is.'" Another teacher says: "When you look in a microscope at a cell and the cell gets bigger and bigger, you are learning that you can see more structure when you change the scale. With the walk-through computer, you get a keyboard big enough to sit on. For these kids, it's just part of taking for granted that you can make a computer bigger and bigger but that doesn't mean that you can see it better."

At the MassCUE discussion, one currently popular position about computer literacy is underrepresented. This is the view that computer literacy should no longer be about the computer at all but rather about the application programs you can run on it. The arguments for this position are strong. One is grounded in practical, economic concerns. Entering today's workforce requires fluency with software. Word processors, spreadsheets, databases, internet search engines, computer-aided design programs—these are the tools of contemporary trades. Learning to use these tools demands a new kind of craftsmanship, one that confers a competitive edge. Additionally, like all craftsmanship, there is a thin line between craft and artistry. These tools, artfully used, enable users to discover new solutions to old problems and to explore problems that were never previously envisaged.

Another argument for software fluency as an educational goal goes beyond such practicalities to a more philosophical point. The computer is a simulation machine. The world of simulation is the new stage for playing out our fantasies, both emotional and intellectual. The walk-through computer is its theater, its perfect icon. From this point of view, what children need to know is how to play on this new stage, how to sort out the complex relationship between the simulated and the "real," between representations of the world and the world itself. The "hands-on" manipulation of software may bring these heady issues down to earth. An eleven-year-old child who spends an afternoon manipulating images on Adobe *Photoshop*, creating landscapes that exist only within the computer, may use the software as an object-to-think-with for thinking through issues at the center of contemporary cultural debate. And yet it is often the case—too often the case—that experiences with simulation do not open up questions but close them down.

Simulation and its Discontents

In the 1980s, the controversy in the world of computers and education was about whether computer literacy should be about programming. Would an emphasis on programming skills in the curriculum teach something important, or would it, as some feared in the parlance of the time, turn children into "linear thinkers"? Today, the debate about computers in education centers around the place of educational software and simulations in the curriculum.

"Your orgot is being eaten up," flashes the message on the screen. It is a rainy Sunday afternoon and I am with Tim, 13. We are playing *SimLife*, Tim's favorite computer game, which sets its users the task of creating a functioning ecosystem. "What's an orgot?" I ask Tim. He doesn't know. "I just ignore that," he says confidently. "You don't need to know that kind of stuff to play." I suppose I look unhappy, haunted by a lifetime habit of not proceeding to step two before I understand step one, because Tim tries to appease me by coming up with a working definition of orgot. "I think it is sort of like an organism. I never read that, but just from playing, I would say that's what it is."

A few minutes later the game informs us: "Your fig orgot moved to another species." I say nothing, but Tim reads my mind and shows compassion: "Don't let it bother you if you don't understand. I just say to myself that I probably won't be able to understand the whole game any time soon. So I just play." I begin to look through dictionaries in which orgot is not listed and finally find a reference to it embedded in the game itself, in a file called READ ME. The text apologizes for the fact that orgot has been given several and in some ways contradictory meanings in this version of *SimLife*, but one of them is close to organism. Tim was right—enough.

Tim's approach to *SimLife* is highly functional. He says he learned his style of play from video games: "Even though *SimLife*'s not a video game, you can play it like one." By this he means that in *SimLife*, like video games, one learns from the process of play. You do not first read a rule book or get your terms straight. Tim is able to act on an intuitive sense of what will work without understanding the rules that underlie the game's behavior. His response to *SimLife*—comfort at play, without much understanding of the model that underlies the game—is precisely why educators worry that students may not be learning much when they use learning software.

Just as some teachers do not want to be "reduced" to instructing children in a computer "appliance," many resent providing instruction in a learning environment that often strikes them as an overblown video game. The question of simulation is posed from preschool through the college years. Why should four-year-olds manipulate virtual magnets to pick up virtual pins? Why should seven-year-olds add virtual ballast to virtual ships? Why should fifteen-year-olds pour virtual chemicals into virtual beakers? Why should eighteen-year-olds do virtual experiments in virtual physics laboratories? The answer to these questions is often: because the simulations are less expensive; because there are not enough science teachers. But these answers beg a large question: Are we using computer technology not because it teaches best but because we have lost the political will to fund education adequately?

Even at MIT, the effort to give students ready access to simulation tools has provoked an intense and long-lived debate. In the School of Architecture and Planning, for example, there was sharp disagreement about the impact of computer-aided design tools. Some faculty said that computers were useful insofar as they compensated for a lack of drawing skills; others complained that the results had a lower aesthetic value, making the architect more of an engineer and less of an artist. Some claimed that computers encouraged flexibility in design. Others complained that they made it easier for students to get lost in a multitude of options. Some faculty believed that computer-aided design was producing novel solutions to old problems. Others insisted that these solutions were novel and sterile. Most faculty agreed that the computer helped them generate more precise drawings, but many described a loss of attachment to their work. One put it this way:

> I can lose this piece of paper in the street and if [a day later] I walk on the street and see it, I'll know that I drew it. With a drawing that I do on the computer . . . I might not even know that it's mine.

Another architecture professor felt that simulation not only encourages detachment from one's work, but detachment from real life:

> Students can look at the screen and work at it for a while without learning the topography of a site, without really getting it in their head as clearly as they would if they knew it in other ways, through traditional drawing for example. . . . When you

draw a site, when you put in the contour lines and the trees, it becomes ingrained in your mind. You come to know the site in a way that is not possible with the computer.

In the physics department, the debate about simulation was even sharper. Only a small subset of real-world physics problems can be solved by purely mathematical, analytical techniques. Most require experimentation in which one conducts trials, evaluates results, and fits a curve through the resulting data. Not only does the computer make such inductive solutions easier, but as a practical matter, it also makes many of them possible for the first time. As one faculty member put it:

> A student can take thousands of curves and develop a feeling for the data. Before the computer, nobody did that because it was too much work. Now, you can ask a question and say, "Let's try it." The machine does not distance students from the real, it brings them closer to it.

But Victor Weisskopf, an emeritus professor who had for many years been chair of MIT's physics department, provided a resonant slogan for the anti-computer group. When colleagues showed him their computer printouts, Weisskopf was fond of saying, "When you show me that result, the computer understands the answer, but I don't think you understand the answer." Physicists in the anticomputer camp speak reverently of the power of direct, physical experiences in their own introductions to science, of "learning Newton's laws by playing baseball." For one, simulation is the enemy of good science. "I like physical objects that I touch, smell, bite into," he said. "The idea of making a simulation . . . excuse me, but that's like masturbation."

There is general agreement that since you can't learn about the quantum world by playing baseball, only a computer simulation can provide visual intuitions about what it would look like to travel down a road at nearly the speed of light. But beyond that, simulations are controversial. The pro-simulation faculty stresses that computers make it possible to play with different parameters and see how systems react in real time, giving students an experience of "living physics," but the opposing camp thinks that using simulation when you could directly measure the real world is close to blasphemy. One puts it this way:

> My students know more and more about computer reality, but less and less about the real world. And they no longer even really know about computer reality, because the simulations have become so complex that people don't build them any more. They just buy them and can't get beneath the surface. If the assumptions behind some simulation were flawed, my students wouldn't even know where or how to look for the problem. So I'm afraid that where we are going here is towards *Physics: The Movie*.

Readership in a Culture of Simulation

Of course, both sides of the debating faculty at MIT are right. Simulations, whether in a game like *SimLife* or in a physics laboratory or computer-aided-design application, do teach users how to think in an active way about complex phenomena as dynamic, evolving systems. And they also get people accustomed to manipulating a system whose core assumptions they may not understand and that may or may not be "true." Simulations enable us to abdicate authority to the simulation; they give us permission to accept the opacity of the model that plays itself out on our screens.

Writing in this journal ["Seductions of Sim: Policy as a Simulation Game," Spring 1994], Paul Starr has pointed out that this very abdication of authority (and acceptance of opacity) corresponds to the way simulations are sometimes used in the real worlds of politics, economics, and social planning. Perhaps screen simulations on our personal computers can be a form of consciousness-raising. Starr makes it clear that while it is easy to criticize such games as *SimCity* and *SimHealth* for their hidden assumptions, we tolerate opaque simulations in other spheres. Social policymakers regularly deal with complex systems that they seek to understand through computer models that are used as the basis for actions. Policymaking, says Starr, "inevitably re[lies] on imperfect models and simplifying assumptions that the media, the public, and even policymakers themselves generally don't understand." He adds, writing about Washington and the power of the Congressional Budget Office, America's "official simulator," "We shall be working and thinking in *SimCity* for a long time." So, simulation games are not just objects for thinking about the real world but also cause us to reflect on how the real world has itself become a simulation game.

The seduction of simulation invites several possible responses. One can accept simulations on their own terms, the stance that Tim encouraged me to take, the stance that Starr was encouraged to take by Washington colleagues who insisted that even if the models are wrong, he needed to use the official models to get anything done. This might be called simulation resignation. Or one can reject simulations to whatever degree possible, the position taken by the MIT physicists who saw them as a thoroughly destructive force in science education. This might be called simulation denial.

But one can imagine a third response. This would take the cultural pervasiveness of simulation as a challenge to develop a new social criticism. This new criticism would discriminate among simulations. It would take as its goal the development of simulations that help their users understand and challenge their model's built-in assumptions.

I think of this new criticism as the basis for a new class of skills: readership skills for the culture of simulation. On one level, high school sophomores playing *SimCity* for two hours may learn more about city planning than they would pick up from a textbook, but on another level they may not know how to think about what they are doing. When I interview a tenth grader named Marcia about *SimCity*, she boasts of her prowess and reels off her "top ten most useful rules of Sim." Among these, number six grabs my attention: "Raising taxes always leads to riots."

Marcia seems to have no language for discriminating between this rule of the game and the rules that operate in a "real" city. She has never programmed a computer. She has never constructed a simulation. She has no language for asking how one might write the game so that increased taxes led to increased productivity and social harmony. And she certainly does not see herself as someone who could change the rules. Like Tim confronted with the orgot, she does not know how to "read" a simulation. Marcia is like someone who can pronounce the words in a book but doesn't understand what they mean. She does not know how to measure, criticize, or judge what she is learning. We are back to the idea over which the MassCUE teacher stumbled when trying to describe the notion of an "appropriate" level at which to understand computers and the programs that animate them. When Oliver Strimpel talked about wanting to use the computer museum as a place to teach the power of a transparent understanding of the layers of the machine, he was talking about understanding the "naked" computer. As we face computers and operating systems of an increasingly dizzying size and complexity, this possibility feels so remote that it is easy to dismiss such yearnings as old-fashioned. But Marcia's situation—she is a fluent "user" but not a fluent thinker—re-poses the question in urgent terms. Marcia may not need to see the registers on her computer or the changing charges on a computer chip, but she needs to see something. She needs to be working with simulations that teach her about the nature of simulation itself, that teach her enough about how to build her own simulation that she becomes a literate "reader" of the new medium.

Increasingly, understanding the assumptions that underlie simulation is a key element of political power. People who understand the distortions imposed by simulations are in a position to call for more direct economic and political feedback, new kinds of representation, more channels of information. They may demand greater transparency in their simulations; they may demand that the games we play (particularly the ones we use to make real-life decisions) make their underlying models more accessible.

We come to written text with centuries-long habits of readership. At the very least, we have learned to begin with the journalist's traditional questions: who, what, when, where, why, and how. Who wrote these words, what is their message, why were they written, how are they situated in time and place, politically and socially? A central goal for computer education must now be to teach students to interrogate simulations in much the same spirit. The specific questions may be different but the intent is the same: to develop habits of readership appropriate to a culture of simulation.

Walt Whitman once wrote: "There was a child went forth every day. And the first object he look'd upon, that object he became." We make our technologies, our objects, but then the objects of our lives shape us in turn. Our new objects have scintillating, pulsating surfaces; they invite playful exploration; they are dynamic, seductive, and elusive. They encourage us to move away from reductive analysis as a model of understanding. It is not clear what we are becoming when we look upon them—or that we yet know how to see through them.❑