TRUTH, BEAUTY AND THE VIRTUAL MACHINE

BY DAVID GELERNTER / ILLUSTRATIONS BY MALCOLM TARLOFSKY

WITH EACH PASSING DAY OUR FABULOUS SOFTWARE CREATIONS—OUR VIRTUAL MACHINES—GROW MORE COMPLEX, MORE POWERFUL, AND MORE UNWIELDY. THEY WILL NEVER CARRY US TO A GOLDEN FUTURE UNLESS WE START TO CRAFT THEM WITH BEAUTY AS WELL AS BRAWN.

THE SENSE OF BEAUTY IS A TUNING fork in the brain that hums when we stumble on something beautiful. We enjoy the resonant hum and seek it out. And when we return numb and weary from a round of shoveling the grim gray snow of life, beauty is the hearth, beauty's the fire, beauty's the cup of coffee (the fragrance, the saucer's clink, the curl of cream) that makes the whole business seem almost worthwhile. Strangely enough, beauty is also a truth-and-rightness meter, and science and technology could not exist without it. Its tuning-fork hum guides scientists toward truth and technologists toward stronger and more useful machines. It leads the way forward.

There is the ever-present danger when you discuss beauty in science, mathematics, and technology that readers will assume the word is being used metaphorically. Could a mathematical proof, scientific theory, or piece of software be "beautiful" in the real, literal way that a painting or symphony or rose can be beautiful?

Yes.

Adapted from David Gelernter's forthcoming book, *Machine Beauty: Elegance and the Heart of Computing*, which will be published in January by Basic Books.

The beauty of a proof or machine lies in a happy marriage of simplicity and power-power meaning the ability to accomplish a wide range of tasks, get a lot done. The power-and-simplicity criterion applies to birch-bark canoes, suspension bridges, programming languages, scientific theories, and machines of all kinds. I call this type of beauty "machine beauty"; there is always a happy couple (power married to simplicity) at the heart of it. Not every marriage is happy. Bringing power and simplicity to bear doesn't guarantee machine beauty-just makes it possible, and nothing else does.

But machine beauty bothers us. We act as a society as if our goal were not to nurture or celebrate it but to stamp it out. And our perversity has consequences. We give our scientists and technologists wrong training. We guide our technology enterprises badly. We force some of our most promising new technologies to crawl through bizarre obstacle courses on their bellies instead of greeting them with roses. We tolerate junk cheerfully-in the form of commercial software, for example-that hurts our productivity and adds nuisance to our lives. A "paradox of beauty," in short-we know that machine beauty is crucially important, but it kills us to say so.

Why? That is a hard question and there is no single answer. But at base, machine beauty rubs us wrong. It confuses us and doesn't seem to fit. We still worship science and technology-although we are rarely honest or clearheaded enough, nowadays, to own up to the fact, and to our minds science is objective, logical, analytic, austere, esoteric, highly specialized, and masculine. That our best scientists and technologists are guided by aesthetics is embarrassing. Insisting that beauty is at the heart of science and technology is like tacking ruffles to your office furniture-it takes a serious proposition and makes it frilly and frivolous.

Some inspired technologists achieve beauty on their own. Some push their designs to a certain point, find them aesthetically wanting, and call in designers to finish the job. The designers of the Gestetner duplicating machine made it work but failed to make it beautiful. So in 1929 Sigmund Gestetner looked up young Raymond Loewy in Manhattan and asked him to redesign the awkward, busy-looking duplicator; and by the way, he was sailing for England in three days, so Loewy had better step on it.

A design emerged right on schedule three days later and remained in production, basically unchanged, for 40 years. Loewy's machine was sleek and neat and simple. And it functioned better than the original: worked better as office furniture. The old machine had flared-out legs. They were a hazard; people tripped over them. Loewy straightened them. Obvious change? Certainly, in retrospect. An old photograph shows Gestetner and Loewy side by side as the client inspects the clay model for the first time. Gestetner's smile is unrehearsed-radiant and delighted. The photo is strangely moving.

Loewy's firm designed cars for Studebaker. As usual, aesthetics and function improved simultaneously and were jumbled up in Loewy's thinking: "Weight is the enemy... whatever saves weight saves cost. The car must look fast, whether in motion or stationary."

In artistic terms, the sort of industrial-design beauty we are discussing may sound shallow. Sometimes it is but sometimes not. One of the finest artworks of the century is the 1938 J3 steam locomotive by Henry Dreyfuss for the New York Central's 20th Century Limited. It has a great smooth hemispherical nose divided vertically by a fin down the middle, an elegant smooth cowl shielding the cowcatcher and pilot, solid driving wheels with punched-out holes for counterbalancing; every line serves a purpose, every detail is an indispensable part of the balanced whole, and the finished product has the loveliness of overwhelming power understated. The J3 was sophisticated technology and it was beautiful. It remains beautiful.

In science, beauty's role is less visible but at least as important as it is in technology. Roger Penrose, the distinguished mathematical physicist, writes of the key idea in Kurt Godel's famous theorem that it is "simple, beautiful, and profound." "To anyone who is motivated by anything beyond the most narrowly practical," writes the physicist J. R. Pierce, "it is worthwhile to understand Maxwell's equations simply for the good of his soul." Richard Feynman introduces a mathematical excursus in his physics text: "We could bring forth this formula in two minutes or so... But science is as much for intellectual enjoyment as for practical utility, so instead of just spending a few minutes on this amazing jewel, we shall surround the jewel by its proper setting."

By insisting on the importance of aesthetics, do we ruin science by replacing its strong, hard foundations with soft, squishy ones? No. We say that beauty is subjective, and that's true up to a point but false beyond, and most of the action takes place beyond.

Science, first of all, is indeed a strong, hard, objective business, and in some ways fundamentally different from art. It has become fashionable in certain academic circles to hold that the truths of science are "socially constructed"-not absolutely true, only held to be true by the local authorities, dollars backed by official proclamation and not gold.

Despite the antiscience cult's small size-it is probably no stronger on U.S. campuses than witchcraft or astrology-every scientist (and scholar and artist and citizen) has a duty to repudiate it. That the truths of science are "socially constructed" is false. Any child can see it is false; can see that the wavelength of blue light is the same in New York and Nigeria, that the valence of the oxygen in your body is the same whether you are male or, female, that sine waves were shaped like sine waves yesterday afternoon and ten centuries ago. Of course, you could argue that the whole idea of wavelength is a cultural creation and that some people-ninth-century Catalonian wheel-

Mathematics is serious, aesthetics not; hence computer science has been banging its head against the wall since the mid-1970s in an effort to put programming on a mathematical basis, and made such astonishingly wrights, deconstructionist celebrity English professors-prefer to understand blue light in a wavelength-independent way, or not at all. But if so, they will have a hard time making sense of why the sky is blue, what the redshifted spectrum of distant stars means, or why it is reasonable, given that we can see red and blue light, that we can also see orange but not radar or microwaves.

But what about beauty? Beauty must be socially constructed, right? If you think so, you are forced to play down beauty's importance to science, because science is a unified, coherent whole. It coheres over centuries and millennia. We have relativistic quantum mechanics but still teach Newtonian mechanics too. We have modern algebra but still need the old-fashioned kind. Mathematicians and scientists pose questions that remain interesting and are finally solved centuries later. Modern science incorporates discoveries going back to ancient Greece. If I believe that the beauty sense is a mere social construction, tumbled like trash before the passing breeze of fashion, I am forced to deny that it could possibly have inspired any such stunningly consistent, coherent intellectual structure as science or mathematics.

Many academics do believe, nowadays, that the beauty sense is "socially constructed," and many nonacademics agree-but that claim is largely false also. Fashions change, obviously, and tastes differ. And, yes, the rich, powerful, and prominent swing a lot of weight in the fashion department-always have and always will. Nothing new there. Here is what is surprising: picture a smelly, bedraggled thirteenth-century French mason and his staggeringly foreign world-dangerous, ignorant, and wretchedly poor, muddy and filthy and sick, where there is no such thing as physiology, diatonic music, or freedom; and life is tough and short even for the biggest cheeses. And yet that thirteenth-century French mason finds Chartres cathedral compellingly beautiful (let's say he designed some of it), and so do we. Could anything be more amazing? You could make the same observation about the Book of Samuel or the ***Iliad*** or the Ryoan-ji garden in Kyoto or a crowd of other masterpieces-and you would be struck in each case, I think, not by how greatly but how little beauty depends on social context.

BEAUTY IS CRUCIAL to software also. Most computer technologists don't like to discuss it, but the importance of beauty is a consistent (if sometimes inconspicuous) thread in the software literature. Beauty is more important in computing than anywhere else in technology. And where computers are concerned, the beauty paradox is especially acute.

Beauty is important in engineering terms because software is so complicated. Complexity makes programs hard to build and potentially hard to use; beauty is the ultimate defense against complexity. Beauty is our most reliable guide, also, to achieving software's ultimate goal: to break free of the computer, to break free conceptually. Software is stuff unlike any other. Cyberspace is unlike any physical space. The gravity that holds the imagination back as we cope with these strange new items is the computer itself, the old-fashioned physical machine. Software's goal is to escape this gravity field, and every key step in software history has been a step away from the computer, toward forgetting about the machine and its physical structure and limitations-forgetting that it can hold only so many bytes, that its memory is made of fixed-size cells, that you refer to each cell by a numerical address. Software needn't accept those rules and limitations. But as we throw off the limits, what guides us? How do we know where to head? Beauty is the best guide we have.

What is software? A running program is a kind of machine-a strange kind that gets power and substance from another machine, namely, the computer itself. An executing program is a machine that has been "embodied" by a computer in roughly the sense that a hand puppet is embodied when you slip your hand in. A nonexecuting program is the limp puppet without a hand, an empty shell. Slip a computer inside and it becomes a working software machine: an electric-powered information-transforming machine-in the sense that a clothes washer is an electric-powered clothes-transforming machine.

A running program is often referred to as a virtual machine-a machine that doesn't exist as a matter of actual physical reality The virtual machine idea is itself one of the most elegant in the history of technology and is a crucial step in the evolution of ideas about software. To come up with it, scientists and technologists had to recognize that a computer running a program isn't merely a washer doing laundry. A washer is a washer whatever clothes you put inside, but when you put a new program in a computer, it becomes a new machine.

The virtual machine idea clarifies an important problem-what exactly do programmers do? What activity are they engaged in when they make a program? Are they technical writers? Are they mathematicians? Neither: They are machine designers. They need talent and training of the sort that makes for structural engineers, automobile designers, or (in a general way) architects-not for writers or mathematicians. A program is a blueprint for a virtual machine-a blueprint that gets converted into the thing itself (the executing program, the "embodied" virtual machine) automatically when you hand it to a computer.

Here is a first entry in my guide to the nonexistent Museum of Beautiful Computing:

The virtual machine: A way of understanding software that frees us to think of software design as machine design.

Beauty is decisively important to computer technologists because, first, virtual machines are always in danger of drowning in complexity. Hardware machines are held in check by physical reality. Allow such a machine to get too complicated and no one will be able to afford it, or it will be so heavy it will stave in the floor, or use so much power it will bum up. But software builders don't need to assemble materials or worry about power supplies, heat dissipation, weight, drag, toxicity. So they go wild; a single programmer alone at his keyboard can

li'ttle progress you'd imagine it would have drawn certain conclusions.

provise software machines of fantastic or even incomprehensible complexity. Imagine what kind of palaces people would live in if all you needed to do were to draw a blueprint, hand it to a machine, and see the structure realized automatically at the cost of a few drips of electricity. The most complex machines in the world today are made of software, and the "average" software machine-the typical word processor or spaceship game or operating system-is enormously complicated, too. A modern TV may contain half a million bytes of software (a byte is the size of a single alphabetical character inside the computer); a typical modern car has a 30,000-line program inside.

This huge complexity is responsible for software's permanent crisis: if you build a big enough program, it is almost impossible to make it come out right. Studies show that the average commercial software project takes 50 percent longer than it was supposed to, and one project in four is abandoned. Your only hope is to keep the number of serious bugs low enough so that your program is more or less okay most of the time.

A new airport is scheduled to open in Denver in the fall of 1993; fall '93 comes and goes, months pass, and it is still closed-because the software that is supposed to control the baggage-delivery system doesn't work. A Defense Department satellite tumbles into oblivion because of buggy software. In 1987 an announcement is made in California that two existing, correctly working programs will be merged-the driver registration and car registration systems-and some components added to allow people to use the finished product directly from kiosks. The new system is supposed to be finished by 1993. Then it is supposed to be finished by 1998. Then it is canceled, six years and 40-some-odd million tax dollars after work began, because the task turns out to be in effect impossible. There are many similar stories.

To get out of the crisis, two steps are necessary: programmers need to be better trained, and software builders need to concentrate on making reusable blueprints and frameworks instead of reusable little pieces.

The first issue comes down in significant part to aesthetics. A good programmer can be a hundred times more productive than an average one, easily. The gap has little to do with technical or mathematical or engineering training, much to do with taste, good judgment, aesthetic gifts-and also, to be fair, a quality that has nothing to do with aesthetics: sheer intellectual aggressiveness. And brains don't hurt. But the fact that software's biggest hits are exactly the systems that are repeatedly praised for elegance-the Algol 60 language, from which so much modern practice derives; the "object oriented" programming technique that emerged from Algol in 1967; the Apple desktop, on which the vast majority of computer users rely, in one form or other, today-ought to be a clue to the flummoxed industry that elegance has something to do with good software, that there is a connection somewhere between aesthetics and success.

BUT THE BEAUTY paradox is such that the industry and computer science researchers would far rather pursue mathematical solutions, so-called formal methods, than teach programmers about beauty. Mathematics is serious, aesthetics not; hence the field has been banging its head against the wall since the mid- 1970s in an effort to put programming on a mathematical basis, and made such astonishingly little progress you'd imagine it would have drawn certain conclusions. But when mathematical methods fail, the invariable response is "Bring on more mathematical methods!" A little progress has been made here and there, and mathematics is fine in its place. But it cannot be the whole story or even the main one, or we would not be stuck where we are, in a permanent mudbank spinning our wheels. "The hell with mathematics; let's teach our programmers about beauty" is what we ought to hear. Instead we are solemnly informed (in a representative ***Scientific American*** piece) that "intuition is slowly yielding to analysis as programmers begin using quantitative measurements... The mathematical foundations of programming are solidifying."

"Anyone who wants to analyze the properties of matter in a real problem," Feynman writes, "might want to start by writing down the fundamental equations and then try to solve them mathematically. [Who needs mere intuition when you can have analysis?] Although there are people who try to use such an approach, these people are the failures in this field [On second thought. . .]; the real successes come to those who start from a physical point of view, people who have a rough idea where they are going and then begin by making the right kind of approximations."

Perhaps there is something to be said for intuition after all. "The German emphasis on calculations," David Billington warns apropos of bridge building, "was a double-edged sword; it forced designers to think rationally, but it also drew them away from forms for which they had no calculations, and thus narrowed the range of structural possibilities."

Not only is your typical software machine hugely complicated on the inside; it offers enormous power and a wide range of functions as well. A taste for beauty is the technologist's most important ally, also, in his struggle to produce software that people are capable of using effectively.

Beauty determines which virtual machines triumph and which are rejected, left to rust like old cars in weedy meadows. Ugly virtual machines waste the underlying computer's power and, vastly more important, the user's time, but a beautiful program hovers nearby like an attentive, unobtrusive British butler. A beautiful program's way of doing things is so close to your own that creative symbiosis develops, a thought-amplifying feedback loop. You have an idea and the machine accommodates it immediately-no back talk, no bargaining. The machine's transparency and willingness might even nudge your thinking a step forward.

The software's role is humble and basically passive, but it can amplify your thought-an important accomplishment. One of the field's foremost visionaries, the inventor in the late 1960s of the mouse and the computer window, ran a laboratory with the strange title Augmentation Research Center. According to Douglas Engelbart, computers are tools for "augmenting human intellect."

They can play that role, however, only to the extent they are beautiful. No creative symbiosis is possible with an ugly virtual machine-with a complex or weak program that forces you to bend to its worldview instead of accommodating yours.

In the computer world, beauty is the most important thing there is. •Z