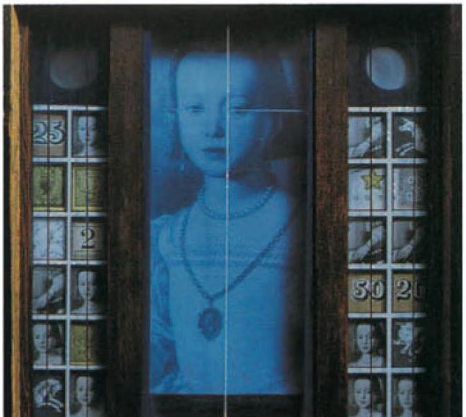
mirror worlds

or: THE DAY SOFTWARE PUTS THE UNIVERSE IN A SHOEBOX...HOW IT WILL HAPPEN AND WHAT IT WILL MEAN



david gelernter

Mirror Worlds

or the Day Software Puts the Universe in a Shoebox . . .
How It Will Happen and What It Will Mean

DAVID GELERNTER
Department of Computer Science
Yale University

New York *Oxford* Oxford University Press

Oxford University Press

Oxford New York Toronto
Delhi Bombay Calcutta Madras Karachi
Kuala Lumpur Singapore Hong Kong Tokyo
Nairobi Dar es Salaam Cape Town
Melbourne Auckland Madrid

and associated companies in
Berlin Ibadan

First published in 1991 by Oxford University Press, Inc.,
200 Madison Avenue, New York, New York 10016

First issued as an Oxford University Press paperback, 1992

Library of Congress Cataloging-in-Publication Data
Gelernter, David Hillel.
Mirror worlds or the day software puts the universe in a shoebox :
how it will happen and what it will mean / David Gelernter.
p. cm. Includes bibliographical references (p.) and index.
ISBN 0-19-506812-2
ISBN 0-19-507906-X (PBK.)
1. Computer software. 2. Software engineering. I. Title.
QA76.754.G45 1991 005—dc20 91-19178

10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

Contents

List of Figures

Acknowledgments

Thanks are due first of all to my graduate students, who made much of this work possible: particularly to Robert Bjornson, Michael Factor, Scott Fertig, Jerry Leichter and Venkatesh Krishnaswamy; to Mauricio Arango and Donald Berndt, for their important contributions to our software; and to Suresh Jagannathan, for countless indispensable hard-core technical discussions.

It takes money to do research: many thanks, accordingly, to Richard Lau of the Office of Naval Research, to Charles Holland of the Air Force Office of Scientific Research, and to the National Science Foundation's Computer Science Program, more or less *in toto*. These three agencies, models of seriousness and of scientific integrity, have championed progressive computing throughout the research community.

Thanks to George McCorkle for valuable suggestions about the manuscript, and for general wisdom and encouragement. However good or bad the final product, it's a lot better than it would have been without his help.

Thanks to Chris Hatchell for providing the administrative glue that holds the whole research effort together, and to Morrow Long of the Yale Computer Science Department and Glenn Fleishman of the Yale Printing Service for their help in producing the manuscript.

This book draws on the fruits of several particularly valuable ongoing collaborations: with Perry Miller and the Medical Informatics Group at the Yale Medical School; with Craig Kolb and Ken Musgrave, of Benoit Mandelbrot's group in the Yale Mathematics Department; with Joe Harris and his colleagues at Sandia National Laboratory in Livermore. Finally, in a category of its own—our ongoing, ever more productive, important and valued collaboration with

Scientific Computing Associates in New Haven.

Thanks to OUP in general and (especially) to my editor Jeff Robbins in particular...

And thanks, finally, to several people who have provided both technical and meta-technical advice and guidance: Steven Nowick, Martin Schultz, Nicholas Carriero, my brother Joel and my father. They have all contributed to this book, and made other contributions besides—*uva'makom she'eiyn 'anashim, hishtadeil lih'yot 'ish.* (In a place where there are no men *you* strive to be a man...) (*Pirkei Avos*, 2:6.) Instead of man, read *mensch*; as a matter of fact, this is what Hillel meant.

Note on the Figures

Figures 5.3 and 5.4 were executed by Michael Factor, Figures 5.1, 5.5 and 5.6 by Craig Kolb, and Figure 7.5 by Jane Gelernter. Figures 5.3 and 5.4 were designed by Michael Factor, and Figure 5.5 by Michael Factor, Craig Kolb and the author. The others are by the author.

Mirror Worlds

This page intentionally left blank

No changing of place
at a hundred miles an hour,

> You can't imagine how strange it seemed
> to be journeying on thus, without
> any visible cause of progress
> other than the magical machine,

nor making of stuffs a thousand
yards a minute,

> with its flying white breath and
> rhythmical, unvarying pace,
> between these rocky walls, which are
> already clothed with moss and
> ferns and grass;

will make us
one whit stronger, happier
or wiser.

> and when I reflected that

There was always more
in the world
than men could see,
walked they ever so slowly;

> these great masses of stone had been cut
> asunder to allow our passage thus far
> below the surface of the earth, I felt that

they will see it no better
for going fast.
JOHN RUSKIN[1] (1856)

> no fairy tale
> was every half so wonderful
> as what I saw.
> FANNY KEMBLE[2] (1830)

This page intentionally left blank

Prologue

This book describes an event that will happen someday soon: You will look into a computer screen and see reality. Some part of your world—the town you live in, the company you work for, your school system, the city hospital—will hang there in a sharp color image, abstract but recognizable, moving subtly in a thousand places. This Mirror World you are looking at is fed by a steady rush of new data pouring in through cables. It is infiltrated by your own software creatures, doing your business.

People are drawn to these software gadgets: When you switch one on, you turn the world (like an old sweater) inside out. You stuff the huge multi-institutional ratwork that encompasses you into a genie bottle on your desk. You can see over, under and through it. You can see deeply into it. A bottled institution *cannot* intimidate, confound or ignore its members; *they* dominate *it*. And your computer's screen is transformed, into a clear surface with brilliant multi-colored life unfolding just beyond. People will stop looking at their computer screens and start gazing into them.

Mirror worlds will transform the meaning of "computer." Our dominant metaphor since 1950 or thereabouts, "the electronic brain," will go by the boards. Instead people will talk about crystal balls, telescopes, stained glass windows—wine, poetry or whatever—things that make you see *vividly*.

Don't like computers? Unamused by technology? For most people, technology is the ocean on a bright cool Spring day. Sparkling in the far distance; breathtakingly cold; exhilarating once you've plunged in. At any rate, not to be over-delicate: This cold and beautiful Ocean is coming to meet you. Mirror Worlds mean another overwhelming rise in sea level. If you don't choose to jump in, what exactly *do* you choose?

Why not give it a try? Hold your breath. Let's plunge.

Chapter 1

Mirror Worlds?

What *are* they?

They are software models of some chunk of reality, some piece of the *real world* going on outside your window. Oceans of information pour endlessly into the model (through a vast maze of software pipes and hoses): so much information that *the model* can mimic *the reality's* every move, moment-by-moment.

A Mirror World is some huge institution's moving, true-to-life mirror image trapped inside a computer—where you can see and grasp it whole. The thick, dense, busy sub-world that encompasses you is *also*, now, an object in your hands. A brand new equilibrium is born.

Suppose you are sitting in a room somewhere in a city, and you catch yourself wondering—*what's going on out there?* What's happening?

At this very instant, traffic on every street is moving or blocked, your local government is making brilliant decisions, public money is flowing out at a certain rate, the police are deployed in some pattern, there's a fire here and there, the schools are staffed and attended in some way or other, oil and cauliflower are selling for whatever in local markets... This list could fill the rest of the book. Suppose you'd like to *have* some of this information. Why? Who are *you* to be so nosy? Let's say you're a commuter or an investment house or a school principle or a CEO or journalist or politician or policeman or even a *mere*, humble, tax-paying citizen. Let's say you're just curious. You want to browse, take in the big picture (it's your city, isn't it?)—form some impression of how well the whole thing is working.

So you build a model. You lay out a detailed map on your living room floor. You add little model buildings and bridges and cars and policemen and so on, and lots of blackboards. On the blackboards you will record information that doesn't correspond to any physical object—the state of the budget, the weather; thousands or maybe millions of other tidbits. The blackboards are scattered all over. Given the blackboards, you don't really *need* the map, the buildings and so on—the city-in-miniature. But you've realized that you'll need some way to *organize* all the data you intend to collect; and a recognizable image is a powerful organizing device. If you dump lots of little cars onto currently slow-go streets, use unfinished buildings to indicate construction projects and so on—you can grasp at least *certain* kinds of data quickly. And you can deploy the blackboards in "reasonable" locations—the stock-prices blackboard in front of the little stock exchange building; which gives you at least some hope of finding them.

You buy some long tables and set them up around your model. You have a few dozen phone lines installed. You hire a bunch of people. Most of them will staff the phones; the rest will maintain the model, moving things around and updating the blackboards to reflect the latest information.

Now you hire several thousand more people (let's say) and send them out into the city. Some are posted permanently at interesting points: you'd like to know traffic conditions *everywhere.* Some are assigned to particular institutions: What are the city council, the board of education, the police department, the mayor's office doing at the moment? How's water quality and the value of city bonds holding up? Your researchers are in constant phone-contact with the staffers back in your living room, passing in the latest data for instant transfer to the model.

Good work. (Glad you didn't just sit there brooding...)

Now, whenever that "*what's going on?*" mood is upon you, you need merely rise from your sofa, glance (languidly) at your model and *you know.*

Yes, *but*—it was difficult and a tad expensive to set up; and how good is it, really? Certainly it's a lot better than nothing. But it's interesting that... If you do the smart thing, chuck the whole set-up and build the model *out of software* instead—

You don't merely get something that's *plausible*—that's achievable, where the hardware version is obviously ridiculous, for most people in any case. More important: the software model is staggeringly more powerful than the best hardware model could possibly be.

The *software* model of your city, once it's set up, will be available (like a public park) to however many people are interested, hundreds or thousands or millions at the same time. It will show each visitor exactly what he wants to see—it will sustain a million different views, a million different focuses on the same city simultaneously. Each visitor will zoom in and pan around and roam through the model as he chooses, at whatever pace and level of detail he likes. On departing, he will leave a bevy of software alter-egos behind, to keep tabs on whatever interests him. Perhaps most important, the software model can remember its own history in perfect detail; and can reminisce pointedly whenever it is asked. And everything is up to date, to the millisecond.

Such models, such *Mirror Worlds*, promise to be powerful, fascinating, and gigantic in their implications. They are scientific viewing tools—microscopes, telescopes—focused not on the hugely large or small, but on the *human-scale* social world of organizations, institutions and machines; promising that same vast *microscopic, telescopic* increase in depth, sharpness and clarity of vision. Such Mirror Worlds don't exist, yet. But most of the necessary components have been designed, built and separately test-fired, and we are now entering the assembly stages that will produce complete (albeit small-scale) prototypes. The intellectual content, the *social* implications of these software gizmos make them far too important to be left in the hands of the computer sciencearchy. The rest of this book explains why.

Sounds Like Fun, but So What?

Software today offers assistance to the specialist (in everybody) — not to the citizen. The mere citizen deals with the increasingly perilous complexity of his government, business, transportation, health, school, university and legal systems unaided. Mirror Worlds represent one attempt to change this state of affairs.

You set up a software mirror wherever you like, then allow some complex real-world system to unfold before it. The software faithfully reflects whatever is going on out front. But this is a three-dimensional kind of reflection: The program reaches out and engulfs some chunk of reality. Like a child-sized play village modeled precisely on a real town and tracking reality's every move, the Mirror World supplies a software object to match and track every real one.

A hospital Mirror World has a software version of every patient, doctor, bed, room—and every abstract entity that's important: cash in the bank, drugs on order and so on. Through permanent sensors and ordinary terminal-based record-keeping, the Mirror World reflects the real one. When a patient is scheduled for surgery, the date is noted in the software *Doppelgänger*. When the patient is transferred to the X unit, the software version is too. The organization's current status is presented in the form of an intricate and constantly-changing *picture* that you explore from your computer screen, skimming the surface or diving deeper as you like.

What's the point? These Mirror Worlds are like regular old-fashioned databases, to some extent. If you need to find Shmoe's salary or Schwartz's social security number, you can look it up. But they go far beyond this. The Mirror World is directly accessible, twenty-four hours a day, to the population that it tracks. You can parachute in your own software agents. They look out for your interests, or gather data that you need, or let you know when something significant seems to be going on. You consult the Mirror World like an encyclopaedia when you need information; you read it like a dashboard when you need a fast take on current status. Fundamentally these programs are intended to help you *comprehend* the powerful, super-techno-glossy, dangerously complicated and basically indifferent man-made environments that enmesh you, and that control you to the extent that you don't control them.

So Mirror Worlds function *in part* as fire walls opposing the onslaught of chaos. But they aren't *mere* fire breaks. They are beer halls and grand piazzas, natural gathering places for information hunters and insight searchers. Most important, they are *microcosms*—intricate worlds come alive in small packages. Whether in the shape of a Victorian winter garden, an electric train layout, a Joseph Cornell shadow-box or a mere three-inch plastic dome with snowflakes softly settling inside, microcosms are intriguing. They show you patterns and help you make discoveries that you'd never have come across otherwise. At their best, they are thought-tools of great power and evocativeness.

But Mirror Worlds aren't happening in a vacuum. Today's world technology scene is no placid pond, exactly. And the technological context isn't incidental to this story: it will figure in every paragraph. So we need to step back a moment before we begin.

1791

People were pretty sure, in 1791, that the industrial revolution had *happened.* It was history.

The world had been transformed. The spinning jenny and the power loom, the coke-fired blast furnace and above all Watt's all-powerful steam engine were ready and waiting. In 1791, William Hutton frothed over in describing the growth of Birmingham, that industrial super-city. "These additions are so amazing, that even an author of veracity will barely meet belief."[1] Fifteen years *earlier*, Adam Smith had enthused over the "universal opulence"[2] that England's state-of-the-art, machine-driven economy had produced. "Let any person consider the progress of everything in Britain during the last twenty years..."[3] wrote Arthur Young in 1774.

In retrospect, little had changed. There were no railroads. Cotton was a minor industry in Britain. Manchester was a smallish town. Who were these people kidding, patting themselves on the back and congratulating themselves up and down on the advanced state of their technology? On the world-transforming wonders it had wrought? In 1791, the industrial revolution was merely building up a head of steam. The Big Bang came later.

Glancing backwards from a vantage-point two centuries hence, 1991 will look a lot like 1791. The technological world of today has that same pastoral sparsely-settled leafiness. Everything is neat and well-ordered and tentative: a garden in earliest Spring. Nothing basic has changed. Yes, software has accomplished great things. But as Al Jolson so presciently announced in Hollywood's first Talkie, you ain't heard nothing yet. The real software revolution won't have much to will likely be the most complex information machines ever built. They might possibly be the most complex machines of *any* kind every built.

But in discussing these prodigies of power and complexity, I'll need to deal, too, with the simplest of information machines.

§ The Paleolithic, Revisited

In the hyper-compressed world of software technology, it's the twenty-first century and the old stone age simultaneously.

At the same time we develop vast complex software worlds, the *simple machines* of information structure are *also* just being invented. The wheel, the ramp, the wedge, the screw, the lever. Much of today's software-structures research amounts precisely to this search for universal, simple information-machines that can support vast complex structures. It makes no sense to reinvent the bolt and the geartrain every time you design a mechanical device. Builders of information machinery too would prefer to start with the universal, basic stuff in hand. But what *are* the simple information machines? It took some time (presumably millenia) to come up with the initial basic five. We'd like five more by this afternoon. If it's not too much trouble.

But you can't merely will them into existence. In some respects, you're faced with a design problem (what engineers do); in others, more a problem of *discovery.* The wheel and the lever are man-made tools. But they're so pervasively important that they look almost like natural phenomena. Not mere engineering doodles; almost *intrinsic* in the logical woodwork of mechanical problem-solving, just waiting to be found out. Chances are, there is a set of basic *information* machines waiting to be found out as well. And Mirror Worlds are *so* complicated that it's especially important that we build them out of simple, universal elements; otherwise, we drown in complexity.

This re-creation of technology on a new footing, in a new way, is an intellectual event of real importance. It's also the sort of event that, by its very nature, is easy to miss. These are *simple* machines we're talking about, after all. Nothing fancy; nothing showy. When the first test-pryings using a prototype lever came off successfully at some Paleolithic research institute (whose funding no doubt had just been slashed)—most likely, no-one held a press conference. A pivoting stick: big deal. Many thousands of years later, early research trials of the potter's wheel in Ancient Sumeria[4] are unlikely to have dominated dinnertable conversation at mid-town watering holes. The Sumerians knew how to write, but this new "wheel" technology hardly ranked up there with cattle inventories as a favorite topic. We are more technology-conscious today. When a rocket goes up, we are moderately interested. Far from the camera crews, technologists are doing experiments that are laughably less spectacular. But, maybe, more important: They are inventing *new* simple machines; recreating the technological universe.

Of course, technology has gone through many phases since the last bout of simple-machine development culminated, with the wheel, in rather a big way. New engineering vocabularies arose for making metal, building structures, harnessing water and wind power, keeping time, building many kinds of engines: a series of techno-revolutions of earthshaking importance. And yet, all these machines were made of *stuff* and, in *one* important sense anyway, an *information machine* is not. The rise of information machinery may or may not prove more important, in the end, than the rise of clocks or watermills or electric power. But it should be clear that *something* is going on here that is new, different and worth understanding. We're starting a new chapter in technological history. Chapter one—machines built out of something; chapter two, machines built out of nothing. Merely *enacted*, temporarily *embodied* by an irrelevant hunk of metal, plastic and silicon called a computer.

And so,

This Book

has three layers. Mirror Worlds don't make sense if they are served in isolation. They are much happier as the insides of a conceptual Deli Sandwich. The bottom layer deals with the basic nature of software and the simple machines we are now in the process of inventing. The upper layer deals with the ultimate motivation in building Mirror Worlds—the search for what I will call "topsight." Each chapter serves up another slice of Mirror World between these wrappers. The goal is not the usual (for this kind of book) quick stroll down Wondertech Boulevard, admiring the window displays and then back tail. Then (Chapter 6) we look at the attic, where bushels of fact are fed into the History Presses ("expert datapools") for transformation into something like "the wisdom of experience." The expert datapools we describe are simple programs with a propensity to behave in complicated ways—to "speculate," to "get distracted," occasionally to "free associate." What would you want with a program that occasionally stops paying attention to you? Good question. I'll get to it...

And finally the Mirror World itself: In Chapter 7 we describe how the whole thing is put together, out of the components and using the technologies we've already explored in some detail.

Discussing Mirror Worlds means discussing the future. But we aren't talking about hazy science fiction. We have arrived at the musing, prototype-building and detailed planning reflected in this book by the fact that the tools and materials for Mirror World building are in hand, and the job is underway.

With infomachinery it's a different story. Programs that amount to a quarter of a million lines of text (there are about 7000 lines in this book, so picture 35 volumes of program) are not in the least unusual. Many programs are much longer. 250,000 lines is enough to create an enormously complex info-landscape with many thousands of regions. How can you design, build and understand such complex landscapes?

Not easily.

It's very hard to make programs come out right. After a decent amount of effort they tend to be *mostly* right, with just a few small bugs. Fixing those small bugs (a bug can be small but catastrophic under the wrong circumstances) might take ten times longer than the entire specification, design, construction and testing effort up to this point. These are *subtle* structures.

If you're a software designer and you can't master and subdue monumental complexity, you're dead: your machines don't work. They run for a while and then sputter to a halt, or they never run at all. Hence "managing complexity" must be your goal. Or, we can describe exactly the same goal in a more positive light. We can call it *the pursuit of topsight.* Topsight—an understanding of the big picture—is an essential goal of every software builder. It's also the most precious intellectual commodity known to man.

§ Engineering Topsight

To manage software complexity, you must seek a deep and thorough understanding of the structure of your problem; and then you must transfer this understanding directly into software. Like studying a face carefully enough to achieve a deep understanding of what it *really* looks like, then transferring this understanding directly into a painted portrait. The goal of the exercise is to achieve something that is so universally important and yet so hard to come by that it doesn't even have a word to describe it. So I'll make one up: *topsight.* (I don't like this coinage particularly and would be glad to have a better one. If you can think of one, let me know.)

If *insight* is the illumination to be achieved by penetrating inner depths, *topsight* is what comes from a far-overhead vantagepoint, from a bird's eye view that reveals *the whole*—the big picture; how the parts fit together. ("Overview" comes fairly close to what I mean. But an "overview" is something you either have or you don't. *Topsight* is something that, like insight, you pursue avidly and continuously, and achieve gradually.)

It is *the* quality that distinguishes genius in any field. (What Newton displayed when he saw planets reeling round the sun and teardrops falling as two pieces of one picture; what Churchill showed when he grabbed for the Dardanelles to break an impasse in France; what Hamlet is transfixed by: the special providence in the fall of a sparrow...) It is the keystone of a beautifully transparent definition of *philosopher*: one who seeks "to transcend the world of human thought and experience, in order to find some point of vantage from which it can be seen whole."[2] But topsight is emphatically *not* a feat for philosophers and geniuses only. Every thinking person aims to achieve it—to understand how the parts relate, how it all adds up. It's not easily won. The fact that we don't even have a *word* for this vital commodity is evidence, more than anything else, of our reluctance (or inability) to teach it. Its significance is denigrated by the run-of-the-mill hacks, bureaucrats and cadres who swing chattering from detail to detail like monkeys in branches, never sensing or caring about the forest in the large. Such people more or less run the world. But all thoughtful people—*most* people, when all is said and done—are born with a powerful inclination to seek this thing—

If you're a software designer, at any rate, your task is hard and clear. When you're presented with a difficult problem, you seek topsight; you use whatever topsight you've achieved as your guide through the treacherous terrain of program building.

It's a tall order, but the alternative is clear-cut: to drown in complexity.

§ Programs illuminated by topsight...

have one unmistakable property: clarity. No down-directed gaze can penetrate an opaque structure. The software designer works *always with the aim of coherence and clarity of statement.* (I will return to this phrase...)

Clarity is marked by three major phenomena. Or, in operational terms, you *get* clarity by applying three principles. Perhaps you build software in a constant blaze of topsightful inspiration, and we're merely characterizing your handiwork: we recognize it by the presence of three attributes. Or maybe you're merely trying to achieve the same affect by humble, serious, meticulous work (good for you—give that guy a cigar); you follow three guiding principles. It amounts to the same thing.

§ The Three-Fold Way to Clarity

In software building, there are three ways. (Actually, I'm not sure whether this is an exhaustive list. I suspect it is, but maybe not. In any event, these are the three that underlie this book.) The three principles are *Recursive Simplicity, Uncoupling* and *Espalier.* I'll introduce the latter two in later chapters, and the first below.

The essence of all three methods is the same; it is *design sense.* This is where engineering comes down to *aesthetic judgement.* To impose clarity upon complexity through deep and careful design-thinking is *the* crucial achievement of the master programmer. I've just noted that the software builder works *always with the aim of coherence and clarity of statement*—this is George Henderson, Art Historian, imagining the unknown master architect of Chartres.[3] The software revolution balances ultimately on a fine point of aesthetics.

This fact bears investigating. I'll return to it.

§ Recursive Simplicity

An object is *recursive* in structure when the whole is structurally identical to its parts—or at least to some of them. You can build a large electronic circuit out of smaller pieces that are *themselves* electronic circuits. You can build a large algebraic expression—something plus something else, times something else—out of smaller pieces that are *themselves* algebraic expressions. You don't build a large toaster out of smaller toasters—recursive structures are uncommon; but: The most important event in the history of software happened somewhere around 1959, when the designers of a programming language called "Algol 60" realized that *you can build a large program out of smaller programs.*

Break out the Dom Perignon!! Why? Because this principle represents such an immense break-through for the *clarity* of information machines. I don't need to understand how a million different structures fit together. I need only master a few, a small handful—for