THE MAGIC OF CODE

How Digital Language Created *and* Connects Our World—*and* Shapes Our Future

SAMUEL ARBESMAN

PUBLICAFFAIRS
New York

PublicAffairs
Hachette Book Group
1290 Avenue of the Americas, New York, NY 10104
www.publicaffairsbooks.com
@Public_Affairs

Printed in the United States of America

First Edition: June 2025

Published by PublicAffairs, an imprint of Hachette Book Group, Inc.
The PublicAffairs name and logo is a registered trademark of the Hachette Book Group.

The Hachette Speakers Bureau provides a wide range of authors for speaking events. To find out more, go to www.hachettespeakersbureau.com or email HachetteSpeakers@hbgusa.com.

PublicAffairs books may be purchased in bulk for business, educational, or promotional use. For more information, please contact your local bookseller or the Hachette Book Group Special Markets Department at special.markets@hbgusa.com.

The publisher is not responsible for websites (or their content) that are not owned by the publisher.

Library of Congress Cataloging-in-Publication Data has been applied for.

ISBNs: 9781541704480 (hardcover), 9781541704503 (ebook)

LSC-C

10 9 8 7 6 5 4 3 2 1

To my parents, who first introduced
me to the magical world of computing

CONTENTS

Introduction

Computing as the Supreme Connection Machine

Computing is a wonder of the world, standing alongside the Great Pyramid and the Hoover Dam as something worth marveling at, our jaws agape and minds expanded. Chances are, though, you don't feel this way. Computers and their powers have receded to a sort of background noise, an ever-present technological scenery. Or we look, many times rightly so, at computers with worry and concern, anxious of their powers and their hold on us. Wonder does poke through from time to time, when we notice that, for instance, artificial intelligences have started to write poetry or can generate images by request. We are then forced to acknowledge that, yes, at least at some level, computers are impressive.

But for me, this impressiveness is not the only—or even the most important—cause for wonder when it comes to computing. In the same way that science fiction is not just about whiz-bang gadgetry or swooping starships but also about engaging with mind-boggling ideas and thought experiments, the wonder of computing is not just the cool technologies we see around us. The wonder is that learning about computers—their nature, their history, and the snaking tendrils of their impact—leads to marveling at basically everything around us. Understanding computers and code and computation—in all their weirdness and delights, features and implications—can lead you to think about so many aspects of our world, from biology and life itself to how we use language and how we think. It is the ultimate connector.

The novelist Richard Powers has described the novel as a "supreme connection machine." Computing can be the same thing: when you take it seriously, it's a massive connective engine to everything around us and what we know, acting like a centripetal force that draws together so much of our knowledge and is cause for a kind of panoptic wonder. While we might relegate computer science to a corner of engineering, it is much bigger and unifying, swirling around, drawing in, and touching every other way of thinking and area of knowledge.

In his book *Philology: The Forgotten Origins of the Modern Humanities*, James Turner claims that, before there were fields of history, linguistics, anthropology, and other domains of the humanities, there was philology, a sort of ur-field, similar to how natural philosophy preceded all of the scientific disciplines that we know and love. Philology was the "king of the sciences," examining the origins and etymologies of words and languages but in the process ranging over linguistics, history, archaeology, literature, theology, art, and more.

Philology was a sort of intellectual attractor, using the study of language to draw a huge number of topics into its orbit. J. R. R. Tolkien was a philologist, and, while no doubt an outlier, *The Lord of the Rings* is the kind of wide-ranging and all-encompassing work you can create if you take your philology seriously. But as the study of different humanistic fields expanded and deepened, there was a fracturing and specialization. As historians, literature professors, and archaeologists each carved out their niche, philology slowly faded away from the humanities.

The study of computing is in many ways similar to philology: computation is the same kind of universal solvent, absorbing and melding with many other domains. Computation is not simply a subdiscipline of mathematics or science but almost a liberal art that can range across and interact with other fields.

This is due to the particular nature of code and software. Computer code is not concrete, it's not steel, but it's also not just text. Software consists of spells of crystallized thought. When a programmer writes code, she constructs a model in her mind of how a system works, or how something in the world works, and embodies that mechanic into computer logic. When my daughter works on rudimentary coding projects at school in her technology classes—programming a digital monkey to collect bananas on the screen, for instance—she has to imagine herself as the machine, determining how it operates and how a loop might work. This process of precipitating whatever was imagined by the programmer into an object of language and action is what brings us into an almost magical realm: for the computer, logic is not inert; it is set to running and is given a life of its own.

In fact, once you scratch on the idea of software as the stuff of language marinated with power—scraping your thumbnail over the gray substance of that lottery ticket of technology—so

much is revealed. If you study computation, you are not simply thinking about the nature of loops, data structures, or databases. By thinking of computation broadly and grappling with its profound ideas, you are able to interrogate so much about the world around us. Software is capable of revealing connections to biology, human language, physics, society, and even philosophy. Massive simulations can be built in silico that span civilizations, contain the interactions of a million stars, or even embody the properties of life itself. The complexities of human communication are revealed in large AI systems that chat with the user or translate from one language to another. When we take this wide view toward the nature of code, we can see that its deep history touches on everything from machines to myth. In fact, code is the embodiment of the fantastical stories we have been telling ourselves for millennia that speak of controlling the world through our words. Code is spellwork and magic made concrete.

When we think of computation, then, we must recognize that it touches on all aspects of the human condition. However, most of computing's implications have occurred in this post-philological fracturing: we employ code for specific uses and in different subspecialties but rarely think of it as part of a unifying, all-encompassing domain of thought. But computation is beyond computer science, or computational biology, or any other field developed by tacking "computational" onto some other word. This broad sense of computation is a smashing together of computer science, the humanities, and the sciences both natural and social.

Think of this book as a guide to broadening our understanding of the wonders and weirdness of computation, as a vehicle for understanding the history and nature of computing and its relationship to us as humans. For only when we understand the true breadth of computing can we figure out how best to incorporate

it into our lives, making it work for us, rather than the other way around.

This will necessarily be a personal and subjective exploration of some of my favorite scenic views, as it were, in the wide world of computing, and is organized into three sections—Code, Thought, and Reality—with each one spiraling outward in impact and implications.

What is in store for the reader? After first exploring the nature of wonder itself in computing, Part I, Code, begins by examining what code is—this strange stuff that we use to control our computers—and how it works. After this exploration into code, we move on to what it means to take the magical nature of computation seriously: If coding really is like sorcery, what does this mean for how we think about computers? Is this a fruitful analogy that can teach us something about both code and our relationship to it?

With this foundation laid for understanding code itself, we move to two different aspects of open-ended software creation. The first focuses on how software is made, specifically open source software. It turns out that the world of open source is a kind of textual tradition, and if we view it this way, we can better understand how computer programs are built. The second aspect of open-ended computing is an exploration of one of the quintessential properties of computer programs: that if you do lots of little calculations and operations over and over, you can get something qualitatively new and beautiful. Specifically, by taking this feature of computers seriously, you can write programs that combine mathematics and computing, and from this seeming simplicity unfurl complex and surprisingly rich microcosms. Entire digital universes emerge from computation, from infinitely deep fractals to tiny computer

programs that generate artworks, each one different every time they are run.

Part II, Thought, begins to examine how we actually use code and how it impinges on our thinking. The first chapter of this section is all about language, both natural and computational. Humans are a linguistic species, but so are machines: we interact with them using a variety of languages, and so understanding the nature of programming languages and the ways that they are both similar to and profoundly different from natural language is vital. I then explore what it means when we are able to let everyone, not just a computational elite, code—when each of us can imagine computer programs, take these ideas in our minds, and encapsulate them in software. This democratization of programming, from spreadsheets to AI tools, changes how each of us can think about our relationship with computers. And finally, in this section's last chapter, I directly confront thought itself: What does it mean to build computational tools for improving our thinking? I look at the interaction between thought—this most human of tasks—and machines and also focus on the new artificial intelligence software that is allowing these interactions to blossom.

Part III, Reality, completes the spiral outward into how code impacts our view of the world and the entirety of the cosmos. I first explore the powers and perils of computational simulation, the process of modeling our world inside computers. The next two chapters examine different aspects of biology: the ways in which computing and biology are similar, different, and increasingly colliding, and then what it might mean to create artificial life and encapsulate features of biology within computers. We then examine glitches and errors in computers, and how this is intertwined with the fact that computers are deeply physical objects; they are not disembodied spirits but machines made of metal and sand. After

this exploration of the physical nature of computing, we finish this section with the simulation hypothesis: What if all of reality is but a computer program, and each of us are part of this massive piece of software? What would this even mean, and what might it mean for our humanity? And can you still learn something interesting about computers and the world, even if you don't take this idea that seriously?

I conclude with a bit of philosophizing, a summation but also a sort of reckoning with how to be human in this age of computation. You will find that throughout the book there is a drumbeat of humility—what we can know and what we cannot, what we can do with machines and what might be forever beyond our grasp—and it also appears in this concluding chapter.

While this book is aimed toward those with no programming experience (yet!), I hope seasoned developers might also get something out of it, both due to its broad scope but also because of what I hope is a somewhat distinctive and novel perspective on the world of computation. Whether or not you've experienced computer code firsthand, my sincere goal is that this book becomes a compass-like guide to this broad humanistic perspective, that there is grandeur in this view of computation.

It is a love letter to the computer, in all its glory and implications.

The biblical scholar and translator Robert Alter has noted that ancient Israel was far from a significant player in its world, surrounded by the major empires of Mesopotamia and Egypt and buffeted by their geopolitical machinations. These empires had a physical splendor—ziggurats, pyramids, temples—that the world of ancient Israel could not hope to match. And yet this tiny country produced some of the most timeless and beautiful literature in all of history.

I wonder if there is a larger point here: an enduring edifice can be built in many ways. Some build pyramids or skyscrapers. Others construct worlds from words. The primacy of text and its subtle sophistication—whether to convey ideas, tell stories, teach lessons, or, now with code, act in the world and process information—is something that we need to rededicate ourselves to. Textual splendor is no less an accomplishment than physical richness. But we need to pay attention to the power of text.

In the book of Genesis, Jacob—soon after fleeing his home to avoid his brother's murderous rage—falls asleep on the way to his mother's family. Upon waking from a dream of prophetic wonder, Jacob declares the awesomeness and majesty of his location, "and I did not know it!"

We are surrounded by the power of code and computation, weaving a future of unparalleled emergent strangeness and wonder. It's time we know it.

Let's begin with wonder.