Alex Groce (agroce@gmail.com), Northern Arizona University

Seymour Papert's *Mindstorms: Children, Computers, and Powerful Ideas* is many things, though most strikingly, seen in the right light, a tragedy: it showed a world on the cusp of a revolution in how we learn and how we live, brought about by the computer. That world did not, so far as I can tell, come to be, though the reasons it did not come to be are not entirely clear. Perhaps the failure was overdetermined; a case where, as Robert Irwin says in *The Arabian Nightmare*, the tragedy "was determined and more than determined. There are always more causes than events." I prefer not to think about it, really.

However, it is not that aspect of the book that I want to focus on in this *Passages* review. The book is widely appreciated for offering a new, Piaget-plus-LOGO-turtles, way of looking at how children learn and what school (or no school) could be like. However there are two less widely-appreciated aspects that specifically have much to offer the software engineer. The first is that *Mindstorms* is a book about the joy of debugging, and the *centrality* of debugging. The second is that *Mindstorms* offers a view, though the name never comes up, of epistemology in the mode of Polanyi as much as in the mode of Piaget. For Papert, this mode of knowing, *personal* and provisional, is mostly concerned with the world at large and physics or mathematics. For the software engineer, however, it is also clearly the only way large, complex, software systems can be known: partially and *personally*.

First, let's talk about debugging. I think one of Papert's most acute criticisms of the educational system is that it is extremely focused on right/wrong answers, and in most cases any wrong answer is deemed equivalent to any other. This is not, of course, how anything much in life works, except maybe being a contestant on a quiz show. It's a convenient fiction that makes it easy to assign grades, and indeed some simple questions do have simple and singular answers. But in life, and especially in programming, it's much less important to get the answer right to start with than to be able to move from a somewhat-right answer to a more right answer. If those answers are to questions about how the world works, or to "what should this novel say" or to "how to be good", then the answers will only approach correctness as a limit. And if the program is much more complex than having a turtle draw a circle, then it too will probably never be right: it will always be a work in progress.

Seen in this light, debugging, however frustrating and seemingly intractable it can be at times, is essential to understanding real programs, which are generally too complex to be understood in a logical, coherent, articulate and complete manner. Papert says: "Errors benefit us because they lead us to study what happened, to understand what went wrong, and, through understanding, to fix it. Experience with computer programming leads children more effectively than any other activity to 'believe in' debugging." The alternative formulation is also true: certain ways of learning as a child will also lead one to believe in debugging in programming. Papert notes that one objection to his approach is that the child programmer "hardly understands at all the complex mechanisms at work behind the scenes whenever a Turtle carries out a LOGO command." But of course this is true of any modern programmer using a modern programming language with myriad layered libraries sitting on top of an operating system of abyssal complexity. Somewhere down there is the machine, but even godbolt.org can only unpeel one layer of the onion. For the most part programming is, like a child's LOGO work, a matter of operating at a level of abstraction that glosses over many a detail.

The emphasis on debugging, and on active engagement with things to be learned, is related to Papert's understanding that "scientific knowledge is more similar to knowing a person than similar to knowing a fact or having a skill." Papert connects this view to some degree to the work of Piaget, but I think for software engineers an equally critical connection, suggested by Papert's phrasing, is to the ideas Michael Polanyi put forth most famously in his book *Personal Knowledge*. Polanyi defends a notion of scientific knowledge that is not purely formal or positivist, but is still valid and immune to solipsism. To connect this to the understanding of computer programs (whether the LOGO workings of a child or the most sophisticated project on GitHub) a core idea of Papert and Polanyi is that it can be true that one truly "knows a great deal" about a program and "understands it", but at the same time be unable to consistently articulate much of that knowledge, or consistently make completely true statements about it. The understanding is, as with riding a bicycle or juggling, partly tacit and active, placing oneself in the "spot" of the balls moving, the bicycle weaving, or the function executing. However, as Papert shows, this is also not a one-sidedly *inarticulate* knowledge. Perhaps the most interesting part of *Mindstorms* is Papert's description of how a proper description of one kind of juggling can aid debugging of the ball-tossing procedure, and reduce the time for a neophyte to master it from two or three hours to perhaps half an hour.

Reading *Mindstorms*, then, is not useful only because it presents a glorious vision of the education of children, or because (at least for me) it evokes deep nostalgia about the "golden age of programming." I fondly recall writing many LOGO programs in my youth (though for me it wasn't so much about the turtle: as Papert notes, *recursion* is "the one idea that is particularly able to evoke an excited response"). Rather, it is because *Mindstorms* is an aid to introspection into how we actually go about writing and debugging complex programs, in some ways unmatched in its explication of the deep roots and experiential nature of that art.

Finally, because for both Papert and Polanyi knowledge is personal, knowledge is intimately linked to *passion*. In the preface to *Mindstorms*, Papert describes *falling in love with* gears as his entry into the world of mathematics. Polanyi presents an epistemology that cannot be reduced to a criteria of verification, whether of testability or of pure logic; instead a vision of reality like a "shirt of flame" and "consumed by devotion" is required For both Papert and Polanyi, there is no "phoning it in" for first-rate creations, whether in child's play or in the construction of scientific theories or great computer programs. As Frost wrote:

Only where love and need are one,
And the work is play for mortal stakes,
Is the deed ever really done
For heaven and the future's sakes.