FORGET FOREIGN LANGUAGES AND MUSIC. TEACH OUR KIDS TO CODE

Illustration: Ronald J Cala II

J. PAUL GIBSON began to teach programming classes for teens out of frustration. A computer scientist at the National University of Ireland, he had by 1998 become shocked at the ineptness of his students. "I was seeing 18- and 19-year-olds having trouble with basic programming concepts that I myself had learned when I was 12," recalls Gibson, who taught himself to code on a Sinclair ZX81. "I realized they hadn't seen any programming in school at all up to that point. So I thought maybe one of the problems we were having is that they were coming to it too late." As word of Gibson's classes spread, primary schools in the Dublin area sought his services too.

By the early 2000s, Gibson was using game-design puzzles to teach rudimentary Java to 8- and 9-year-olds. His success with that age group made him wonder: How young is too young to begin coding?

Gibson had read educational psychology that suggested it was futile to try to reach children who had yet to achieve literacy. But when he made experimental forays into classrooms of 5- and 6-year-olds, the kids grasped his lessons with surprising ease.

With the help of a custom Java applet, he was able to get kindergartners to write a tic-tac-toe program, based on step-by-step rules the students formulated as a group. And using colored balls and string, he taught the kids how to create graph algorithms, an essential component of computer science. "We believe our work shows that you can start teaching computer science before students even know how to read and write," Gibson (who now teaches in France) wrote in a 2012 paper.

"Children aged from 5-11 have so much potential for learning about algorithms and computation that it would be a shame to wait until they are teenagers before we teach them the foundations." That notion is still too radical for most educators. While a popular movement is afoot to teach children, including adolescents, to code —touted by the likes of Bill Gates and basketball star Chris Bosh—few people believe that kindergartners can learn how to bend machines to their will.

The general assumption is that young kids lack the faculties to comprehend a topic as seemingly esoteric as programming.

A sorting game devised by computer scientist J. Paul Gibson to teach the concept of algorithms to children.

But that pessimism is at odds not only with the experiences of Gibson and other pioneering teachers but also with the science of language acquisition. Extensive research has shown that because young brains are so adept at picking up languages, it's best to introduce children to foreign tongues as early as possible. This is why so many ambitious parents are now clamoring for kindergartens that offer intensive Mandarin—they want to give their kids the best possible shot at learning a key language of the Asian century.

What those parents likely don't realize is that the same neural mechanisms that make kids sponges for Mandarin likely also make them highly receptive to computer languages. Kindergartners cannot become C++ ninjas, but they can certainly start to develop the skills that will eventually cement lifelong fluency in code.

And encouraging that fluency should be a priority for American schools, because it is code, not Mandarin, that will be the true *lingua franca* of the future.

Perhaps you remember the turtle. In the early to mid 1980s, the Logo programming language, with its iconic turtle-shaped cursor, was the fad in American elementary schools. By using Logo's simple commands to create intricate graphics, kids were supposed to develop mastery over the Apple IIe's that had begun to appear in their living rooms.

But Logo seldom delivered on its lofty promise. The main problem was not the language itself but the lackluster way in which it was taught: Many instructors simply plopped students in front of computers for an hour a week and hoped for the best.

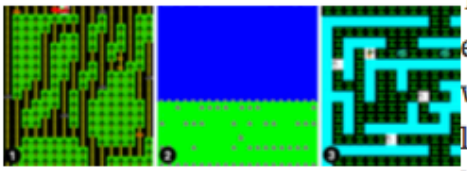
The resulting disillusionment coincided with the emergence of media that transformed school computers from exploratory tools into library aids. "CD-ROMs came out, then the World Wide Web appeared, so you didn't need to know commands to interact with the computer," says Yasmin Kafai, an education professor at the University of Pennsylvania.

Programming vanished from elementary schools for decades, even as computer science became an ever more popular pursuit at the collegiate level. A cultural consensus seemed to spring up: Kids should be taught a nebulous set of "computer skills," but programming—well, that was for grown-ups.

In the past five years, however, a number of groundbreaking projects have begun to prove that consensus wrong. Besides Gibson's tic-tac-toe and graph theory lessons, there is Scalable Game Design, a curriculum developed at the University of Colorado that challenges kids to code their own versions of *Frogger*. At P.S. 185 in Harlem, children as young as 4 are using a language called Cherp to make robots perform household chores. And it's happening overseas too: In Estonia an initiative called ProgeTiiger is striving to teach coding basics to all first graders.

What all these initiatives have in common is an emphasis not on memorizing how to use specific tools but on developing familiarity with the general concepts that underpin all programming—sequencing, conditionals, debugging.

When helping students code their first *Frogger*, for example, Scalable Game Design instructors urge them to think first about the game's various agents—that is, the frog and the lethal vehicles — and then about all the possible interactions those agents can have. The kids slowly learn how to craft rules and conditionals that add up to a logical, functional whole.



Videogames created by grade-school kids through the Scalable Game Design curriculum. Some thoughts from the coders: (1) "The goal is to try to get to the helicopter." (2) "Dodge the rocks and get to the coins." (3) "We are a zombie and we try to reach a diamond. To move you use the arrow keys. You have to not hit the ghost or else you lose."

The fact that young children can manage such elaborate tasks should be no great surprise, given what we know about their knack for acquiring languages. Five-year-olds trump their elders at learning Spanish or Mandarin because young brains are better (so the theory goes) at formulating "procedural" memories—that is, memories that become so deeply embedded in a person's psyche that recalling them is a natural reflex rather than a conscious task.

The evidence is beginning to suggest that as brains age, their capacity for procedural memory diminishes in favor of "declarative" memory, which we use to amass facts. The drawback to declarative memory is that it requires mental exertion to tap into—a huge minus when you're trying to conjugate a tricky foreign verb on the fly. It is far preferable to have those conjugations be second nature to you, as a result of having learned them when your procedural memory was at its sharpest.

No one seems to have researched precisely how programming languages are learned, but there is every reason to believe that they're best absorbed by students primed to form procedural memories.

"I would speculate that the same general-purpose memory systems that underlie language learning in children and adults likely underlie the learning of computer languages," says Michael Ullman, director of the Brain and Language Lab at Georgetown University Medical Center. A key data point in favor of this view is the

evidence

regarding music: great violinists don't start learning the instrument when they're 20 years old but rather when they're 3 or 4, a time when procedural memory is most sensitive.

And what is music if not a form of code—a series of abstract signals that must be sequenced properly in order to please the human ear?

In a perfect world, kindergartners would receive instruction in both programming and foreign language as part of their day. But if a school has to choose, a strong case can be made for code. The most obvious argument, of course, is economic: Demand for software developers already far outstrips supply, and it's expected to increase 30 percent by 2020 — more than double the average for all other jobs. (It's difficult to imagine any scenario in which those opportunities will be outnumbered by jobs requiring fluent Mandarin.)

Yet teaching programming is not just about creating an army of code monkeys for Facebook and Google.

Just as early bilingualism is thought to bring about cognitive benefits later in life, early exposure to coding shows signs of improving what educators call "computational thinking"—the ability to solve problems with abstract thinking. And even for students who never warm to programming, whose innate passions lead them toward English degrees rather than software engineering, understanding code still has great value.

As the media theorist Douglas Rushkoff has observed, to ignore programming is akin to relying on others to drive us around instead of learning to drive ourselves. The majority of our interactions in 50 years won't be with monolingual humans from Asia; they'll be with machines. So let's teach our kids to tell them what to do, rather than the other way around.