21st Century Curriculum | May 2013 Digital Edition

How These Amazing, Kid-Friendly Languages Are Hooking Tomorrow's Programmers

Today's kid-friendly computer languages, like Scratch and Alice, encourage creative expression and exploration.

- By Margo Pierce
- 06/11/13

This article, with an exclusive video interview, originally appeared in T.H.E. Journal's [May 2013 digital edition](#).

Forty years ago, when large mainframe computers roamed the earth, few experts gave much thought to how these mammoth machines could be used for education, and fewer still about how they could help young learners create, explore, and learn through technology. At the time, highly trained programmers still worked in inaccessible languages that mainly processed numbers. But all that changed with a turtle. In 1967, MIT professor Seymour Papert and colleagues developed Logo, an early language for children. Its main innovation? A small robot--the turtle--that students could easily program to move or rotate. For the first time, young programmers got instant feedback and a physical manifestation of their commands.

While Logo's use spread throughout the 1970s, programming never achieved the influence in schools that Papert had envisioned. It wasn't considered a viable educational tool until schools had routine access to computers. Even now, at a time when computers are pervasive in everyday life, many educators still question the value of children becoming articulate in the language of technology--

programming. But as STEM and Common Core concepts--with their emphasis on math, science, and critical thinking skills--begin to shift curricula across the K-12 spectrum, coding is sparking renewed interest.

"We really need to broaden, to rethink what it means to be fluent in today's society," says Mitch Resnick, the LEGO Papert Professor of Learning Research at MIT. "The ability to program, the ability to code, is an important part of being 'fluent' today. In the same way that learning to read opens up opportunities for many other things, and learning to write gives you a new way to express yourself and seeing the world, we see that coding is the same."

In schools where programming is taught, it often acts as a stand-alone class or as part of an after-school program. According to Susan Einhorn, the chair of the management team that runs Papert's company LCSI, part of the reason programming hasn't seen greater integration is that there is no consensus about where it fits within the educational curriculum,. The lack of qualified computer-science teachers and educators comfortable enough with technology to teach programming is another barrier, as is a general resistance to a class that looks more like fun than substance.

"Just because something's fun doesn't make it easy. Seymour would describe it as 'hard fun,'" Einhorn says. "We learn through hard fun. We have to stop seeing learning as rote. It is an active, participatory thing. When you're engaged in something, then you learn the most because you are exploring it."

Papert also contends that people learn better when they're engaged in creating something that is personally meaningful to them. To that end, MicroWorlds, LCSI's current iteration of Papert's programming language for children, encourages curiosity and experimentation beyond the precise syntax and complex character strings demanded by languages like Java and C++. With MicroWorlds and other languages like it, students can drag and drop commands and test their creations without miring themselves in the minutiae of syntax, which can be confusing for both students and teachers.

The strategy isn't new--it was all part of Papert's educational philosophy developed in the time of Logo. "'Constructionism' was a term invented by Seymour Papert," says Einhorn. "It means that, if you're constructing something externally, you help build that knowledge within your head, so that it's not just abstract.… You're also gaining new ideas about how the world works and new understanding."

Coding to Learn

Like Papert, MIT professor Resnick has learned the value of keeping kids interested while teaching them the fundamentals of technology. In 1989, he cofounded Computer Clubhouse, a Boston-based club for a group of kids who were curious about creating with technology but were otherwise underserved by the local community. This experience also underscored the need for a free programming language that was both accessible and capable of helping students create a wide range of projects.

When Resnick first started working with schools, kids weren't using any programming languages to create projects. They used software programs such as Photoshop to create collages, music software to orchestrate compositions, and video programs to bring the different elements together. Contemporary, general-use programming languages were not user-friendly for young people or teachers, and other kid-friendly languages were somewhat limiting, so Resnick and his team decided to create the next generation of constructivist programming language for kids. Their language, Scratch, was released in 2006. Like other languages aimed at kids, Scratch's interface is based on a drag-and-drop, building-block approach that lets users experiment with variables and conditions in an intuitive way.

"We wanted to have a programming language [with which] you could build your projects and your programs by tinkering, the same way you do with LEGO bricks," Resnick says. "That led us to the graphical programming approach that we use."

In the seven years since its release, Scratch has become a community, thanks to social media tools that bring together students from around the world. Today, enthusiasm among teachers is growing as Scratch users share their experiences, lessons, and challenges with each other via ScratchEd, Facebook, and personal connections made at MIT trainings, meet-ups, and other Scratch-based events.

"We felt the best learning experiences happen when kids are interacting and sharing and collaborating with one another," Resnick says. "Our goal is not to help kids to learn to code but code to learn. The coding or the programming is not the end goal; it's more a means of learning many other things."

Tools to Create
Engaging kids on a personal level is part of what's made Joanna Boyd's computer programming class at Bob Miller Middle School in Henderson, NV, so popular that it expanded from a nine-week experiment to an 18-week, project-based course. Boyd uses a number of programming languages designed for young learners, including Scratch and Alice (See the "Learn the Languages" sidebar). She teaches students to develop their project ideas, created on a storyboard, into finished products that they present to the class.

Rather than inventing and assigning new projects for students, Boyd encourages students to create their own programs around what they're already learning in other classes. While some students might create a program around the structure of DNA that they're learning in science class, others take an opportunity to bring their understanding of books like *A Wrinkle in Time* to life. "Now I've got the hook of what they're doing every day, what they're the experts in, and then I'm giving them the tool of Alice to create," says Boyd.

That hook can also provide a level of excitement and engagement that students might not otherwise get from the curriculum. Namely, Boyd says, they're having fun. "I really believe it's exciting their brain--that part of the brain that education doesn't give them," she says. "I think we're so driven by standards and curriculum and teaching to the test that we've lost the creativity of education."

When a student hits a snag in programming, Boyd engages the entire class to figure out how to resolve the problem. Another student might have already encountered the same issue and can walk through the solution. Although she lets the students choose projects that interest them, as the facilitator she provides resources for researching and learning new programming concepts and does what she can to use programming to further their overall education.

"These students do not know how to connect the subject areas with each other. I try to do that in this class. I make it a relationship," Boyd says. "You need to see the whole picture in order to accomplish the task. You need to strategize, and that's what I feel I'm giving at this level, which is a more teachable level."

The result is increased self-esteem, shy children "getting out of their seats," and special-needs children participating with other students. According to Boyd, girls are outperforming boys in mastery of the language and quality of work. She adds that the shared experience of working with technology leads students to connect with their parents in a way she hasn't seen with other subjects. For example, she recalls a father who works at the Hoover Dam bonding with his son over a programming assignment. But it's not all work, all the time.

"Programming gives the students a logical method to present themselves in a creative way and own their learning," Boyd says. "Logic to a lot of people means math, but it really is math being creative. It's about learning and owning it. That's what they do in programming. They get to be logical and creative at the same time."

Learn the Languages
The history of computer languages designed with kids in mind dates back to the 1960s. Here are three popular iterations that students are currently exploring.

Alice: Created as a way to teach programming theory to young students, Alice lets users experiment with 3D animations, games, and videos through drag-and-drop programming of interactions between virtual people and objects in a 3D world--making it especially useful for storytelling exercises.

MicroWorlds: Using the Logo language designed by Seymour Papert, LCSI's flagship software, MicroWorlds EX, is designed for children starting in fourth grade. Still based around--but no longer limited to--Logo's famous turtle, the program lets students command an object, animate it, or have it interact with other objects.

Scratch: Developed by an MIT team led by computer science professor Mitch Resnick, Scratch is a colorful, easy-to-learn programming language used by children as young as 5. Users drag and drop blocks, stacking programming fundamentals, such as conditions and actions, on top of each other to create animations or other types of programs--without regard to syntax and other hallmarks of advanced computer languages.

About the Author

Margo Pierce is a Cincinnati-based freelance writer.