INVITED COMMENTARY

A Different Approach to Coding

Mitchel Resnick, MIT Media Lab, Cambridge, MA, USA

David Siegel, Two Sigma, New York, NY, USA

Over the past few years, there has been a surge of interest in coding (or computer programming). Millions of students have participated in the Hour of Code, an annual effort to encourage schools around the world to spend at least one hour introducing students to coding. And many schools are going beyond these initial coding activities. A growing number of cities and even entire countries are adding coding as a requirement in their school curricula.

We are strong proponents of children learning to code, but we have concerns about the motivations and methods underlying many of these new learn-to-code initiatives. Many of them, motivated by a shortage of programmers and software developers in industry, focus especially on preparing students for computer science degrees and careers, and they typically introduce coding as a series of logic puzzles for students to solve.

We co-founded the Scratch Foundation in 2013 to support and promote a very different approach to coding. For us, coding is not a set of technical skills but a new type of literacy and personal expression, valuable for everyone, much like learning to write. We see coding as a new way for people to organize, express, and share their ideas.

This approach to coding is embodied in our Scratch programming software developed at the MIT Media Lab and available for free online. With Scratch, children ages eight and up snap together graphical programming blocks to create interactive stories and games with animated characters. They can share their projects in the Scratch online community, where others can try them out, give feedback and suggestions, and even revise and extend the projects with their own ideas.

Since the launch of Scratch in 2007, children around the world have shared more than 11 million projects in the Scratch community. More than 17,000 new projects are added each day.

Young people use Scratch in many different settings: homes, schools, libraries, community centers. As they create and share projects in Scratch, they are learning more than coding skills. They are learning to think creatively, reason systematically, and work collaboratively—essential skills for everyone in today's society.

DOI: 10.4018/IJPOP.2015010101

Consider Jocelyn, a teenager who has programmed and shared more than 200 projects over the past two years in the Scratch community under her Scratch username CrazyNimbus. She's developed a wide variety of projects, including a helicopter-flying game (Figure 1), several dress-up doll games (Figure 2), a back-to-school animation with dancing pencils and calculators, and a countdown animation that calculates and displays the number of days until Thanksgiving.

Jocelyn initially came to Scratch to learn how to create games. She stayed because of the community. "I shared my first project, I got instant feedback, and I knew I needed to keep going and keep coding," says Jocelyn.

Figure 1. Helicopter game

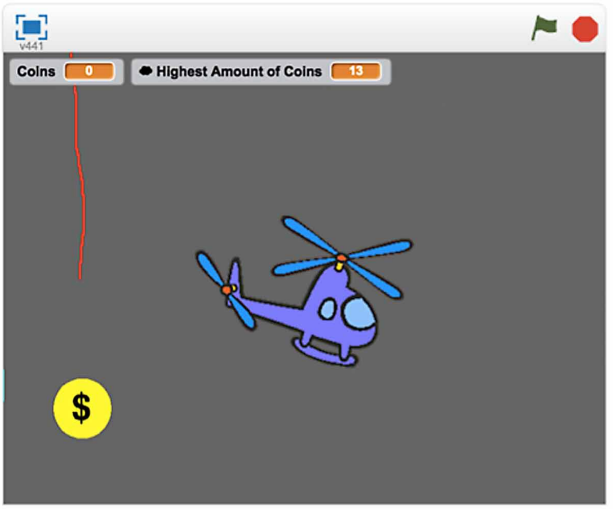
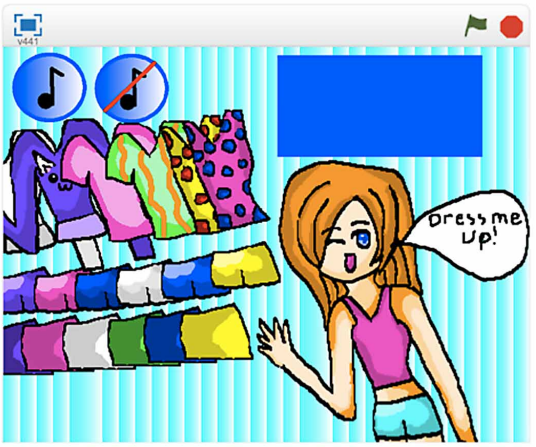


Figure 2. Dress-up game



Scratch is similar to other coding approaches in that it requires systematic, logical reasoning. To create her projects, Jocelyn needed to learn a variety of problem-solving strategies, such as breaking complex problems into simpler parts and continually revising her projects when they didn't work as expected.

But where Scratch is different from the majority of coding approaches is that it puts a high priority on children learning to express themselves creatively and to share their creations with others. For example, Jocelyn used Scratch to program interactive birthday cards for her friends, each with its own personalized animation. Jocelyn also developed an interactive tutorial that showed other children how to make their own customized Valentine's Day cards. "Everyone needs a voice with computers," she says. "I like sharing my projects with the world and seeing what other people think about them."

Most learn-to-code initiatives do not support this type of creative expression. In many introductory coding activities, students are asked to program the movements of a virtual character navigating through a set of obstacles toward a goal. This approach can help students learn some basic coding concepts, but it doesn't allow them to express themselves creatively—or develop a long-term engagement with coding. It's like offering a writing class that teaches only grammar and punctuation without providing students a chance to write their own stories. With Scratch, our goal is for young people to become *fluent* with coding—not only learning the mechanics and concepts of coding, but also developing their own voice and their ability to express their ideas.

As young people express themselves with Scratch, they begin to think of themselves differently, just as they do when they learn to write. The Brazilian educator Paulo Freire recognized that writing is more than just a practical skill. He led literacy campaigns in poor communities not simply to help people get jobs, but to help people learn that, in his words, "they can make and remake themselves."

We see something similar happening when kids code with Scratch. Jocelyn and other Scratch community members begin to see themselves as creators who are able to make their own projects on the computer, not just interact with programs created by others. "It's a life changer," says Jocelyn. In today's society, digital technologies are a symbol of possibility and progress. As young people create and share projects with Scratch, they begin to see the possibility of contributing actively and fully to society.

If coding is going to make a true difference in children's lives, it's important to move beyond the traditional view of coding as simply a technical skill or just a pipeline to getting a technical job. Educators, parents, policymakers, and others should think carefully about their goals and strategies for introducing coding to young people. Based on our research at the MIT Media Lab, we suggest four guiding principles for how to introduce coding:

1. **Projects:** Provide children with opportunities to work on meaningful projects (not just puzzle-solving activities) so they experience the process of turning an initial idea into a creation that can be shared with others.
2. **Peers:** Encourage collaboration and sharing, and help children learn to build on the work of others. Coding shouldn't be a solitary activity.
3. **Passion**: Allow children to work on projects connected to their interests. They'll work longer and harder—and learn more in the process.
4. **Play:** Encourage children to experiment playfully—try new things, take risks, test the boundaries, learn from failures.

By keeping these four Ps in mind, educators and others can make sure that coding lives up to its full potential as a new form of literacy and personal expression—not just another educational fad.

AUTHORS' NOTE

Mitchel Resnick is Professor of Learning Research at the MIT Media Lab. David Siegel is Co-Chair of the algorithmic investment manager Two Sigma. Together, they co-founded the non-profit Scratch Foundation to support creative coding for everyone. A version of this essay previously appeared on Bright.