educational technology®

the magazine for managers of change in education

Volume 52, Number 4 July–August 2012

Point of View

Reviving Papert's Dream

Mitchel Resnick

It has been more than 40 years since Seymour Papert published, in this magazine, his first public article about the Logo programming language. The article, co-authored with Cynthia Solomon, was titled "Twenty Things to Do with a Computer." It described how children might program computers to control robots, compose music, create games, draw recursive pictures, and do many other creative activities.

It was a radical vision. At the time, in 1971, computers still cost tens of thousands of dollars, if not more. The first personal computers would not become commercially available for another five years. Yet Papert foresaw that computers would eventually become accessible for everyone, even children, and he wanted to lay the intellectual foundation for how computing could transform the ways children learn and play.

Some aspects of Papert's dream have become a reality. Children today have access to computation to a degree that few people could have imagined in 1971. Millions of children around the world interact with computation in a wide variety of forms: electronic toys, mobile phones, game machines, laptops, tablets. And they use computational devices to engage in a diverse range of activities: playing games, chatting with friends, exploring virtual worlds, searching for information online.

At the same time, important elements of Papert's dream remain unfulfilled. Papert envisioned a world in which children not only learn to use new technologies, but become truly *fluent* with new technologies. In Papert's view, children should be able to design, create, and express themselves with new technologies. Rather than just interacting with animations, games, and simulations, children should learn to program their own animations, games, and simulations—and, in the process, learn important problem-solving skills and project-design strategies.

Unfortunately, most young people today haven't achieved that type of fluency. Even though young people are sometimes called "digital natives," most use computational devices simply to browse, chat, run apps, and play games. It is as if they can "read" but not "write."

What happened to Papert's dream? When personal computers became available in the late 1970s and early 1980s, there was initial enthusiasm for teaching children how to program. Thousands of schools taught millions of students to write programs in Papert's Logo programming language. But the initial enthusiasm didn't last. Many teachers and students had difficulty learning to program in Logo, since the language was full of non-intuitive syntax and punctuation. To make matters worse, Logo was often introduced through activities that did not sustain the interest of either teachers or students. Many classrooms taught Logo as an end unto itself, rather than as a new means for students to express themselves and explore what Papert called "powerful ideas."

Before long, most schools shifted to other uses of computers. They began to see computers as tools for delivering and accessing information, not for designing and creating as Papert had imagined. Today, most educators view computer programming as a narrow, technical activity, appropriate for only a small segment of the population.

It doesn't need to be that way. In my research group at the MIT Media Lab, we still believe in Papert's dream of computational fluency for everyone. But we also understand that turning Papert's dream into a reality isn't easy. It will require a new generation of technologies, activities, and educational strategies.

That's what we've tried to do with Scratch *(http://scratch.mit.edu)*, a programming language and online community that we developed to provide new pathways into programming. Scratch aims to engage everyone, of all backgrounds and interests, in creating their own interactive stories, games, animations, and simulations. Our goal is not to prepare people for careers as professional programmers but rather to enable everyone to express themselves creatively through programming.

Learning lessons from Papert's experiences of Logo, we've designed Scratch to move beyond Logo along three dimensions, making programming more tinkerable, more meaningful, and more social.

More Tinkerable

For many years, my research group has worked closely with the LEGO Company, helping develop LEGO Mindstorms and other robotics kits. We have always been intrigued and inspired by the way children play and build with Lego bricks. Given a box full of them, children will immediately start tinkering. They'll snap together a few bricks, and the emerging structure will give them new ideas. As they play and build with LEGO bricks, plans and goals evolve organically, along with the structures and stories.

We wanted the process of programming in Scratch to have a similar feel. The Scratch grammar is based on a collection of graphical "programming blocks" that children snap together to create programs (***Figure 1***). As with LEGO bricks, connectors on the blocks suggest how they should be put together. Children can start by tinkering with the bricks, snapping them together in different sequences and combinations to see what happens. Scratch blocks are shaped to fit together only in ways that make syntactic

Mitchel Resnick, *LEGO Papert Professor of Learning Research at the MIT Media Lab, develops new technologies and activities to engage people (especially children) in creative learning experiences. His Lifelong Kindergarten research group developed ideas and technologies underlying the LEGO Mindstorms robotics kits and Scratch programming software, used by millions of young people around the world. He also co-founded the Computer Clubhouse project, an international network of 100 afterschool learning centers where youth from low-income communities learn to express themselves creatively with new technologies. In 2011, he was awarded the McGraw Prize in Education and the World Technology Award in Education (e-mail: mres@media.mit.edu).*

Figure 1. Scratch programming blocks.

sense. There is none of the obscure syntax of traditional programming languages.

The name "Scratch" itself highlights the idea of tinkering, as it comes from the scratching technique used by hip-hop disc jockeys, who tinker with music by spinning vinyl records back and forth with their hands, mixing music clips together in creative ways. In Scratch programming, the activity is similar, mixing graphics, animations, photos, music, and sound.

The scripting area in the Scratch interface is intended to be used like a physical desktop (***Figure 2***). You can even leave extra blocks or stacks lying around in case you need them later. The implied message is that it's OK to be a little messy and experimental. Most programming languages (and computer science courses) privilege top-down planning over bottom-up tinkering. With Scratch, we want tinkerers to feel just as comfortable as planners.

More Meaningful

We know that people learn best, and enjoy most, when they are working on personally meaningful projects. So, in developing Scratch, we put a high priority on supporting a diversity of project genres (stories, games, animations, simulations), so that people with widely varying interests can all work on projects they care about. We also put a high priority on *personalization*—making it easy for people to personalize their Scratch projects by importing photos and music clips, recording voices, and creating graphics.

These priorities influenced many of our design decisions. For example, we decided to focus on 2D images, rather than 3D, since it is much easier for people to create, import, and personalize 2D artwork. While some people might see the 2D style of Scratch projects as somewhat outdated, Scratch projects collectively exhibit a visual diversity and personalization missing from 3D authoring environments.

We continue to be amazed by the diversity of projects

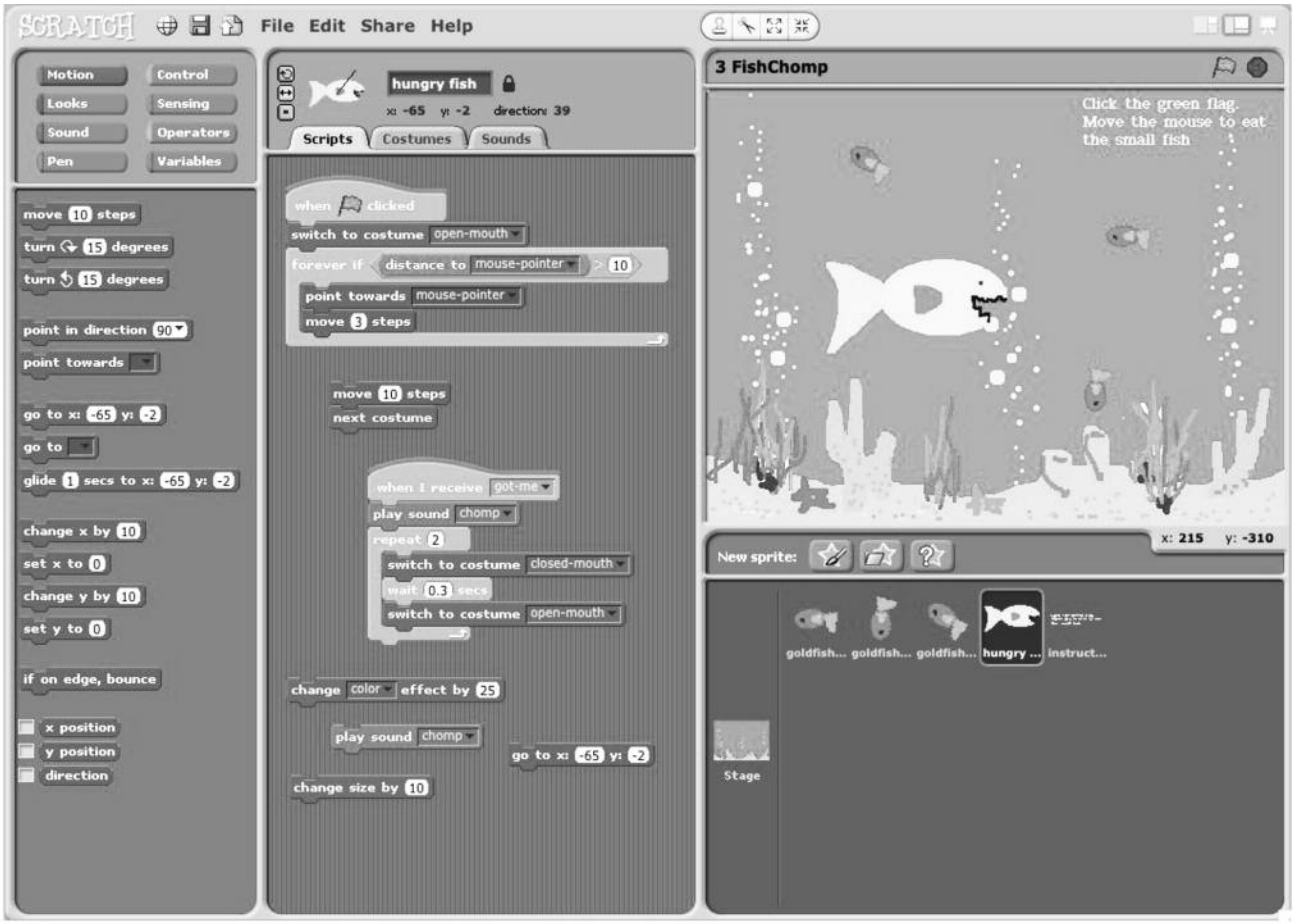

Figure 2. Scratch application interface.

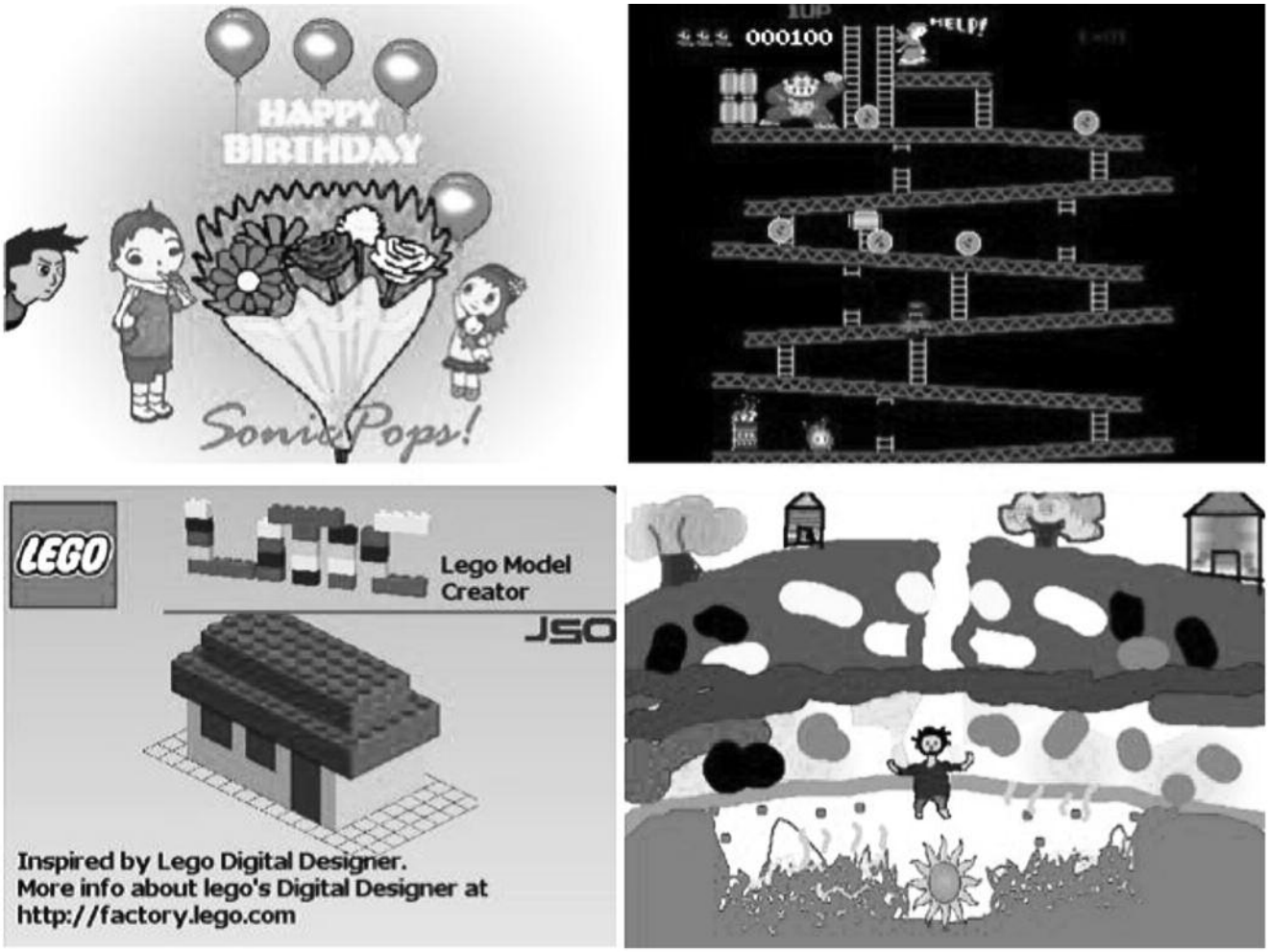

Figure 3. Examples of Scratch projects.

that appear on the Scratch Website (***Figure 3***). As expected, there are lots of games on the site, ranging from painstakingly recreated versions of favorite video games (such as Donkey Kong) to totally original games. But there are many other genres, too. Some Scratch projects document life experiences (such as a family vacation in Florida), while others document imaginary wished-for experiences (such as a trip to meet other Scratchers). Some Scratch projects (such as birthday cards and messages of appreciation) are intended to cultivate relationships. Others are designed to raise awareness on social issues (such as global warming and animal abuse).

As Scratchers work on personally meaningful projects, we find they are ready and eager to learn important mathematical and computational concepts related to their projects. Consider Raul, a 13-year-old boy who used Scratch to program an interactive game in his after-school center. He created the graphics and basic actions for the game but didn't know how to keep score. So when a researcher on our team visited the center, Raul asked him for help. The researcher showed Raul how to create a variable in Scratch, and Raul immediately saw how he could use it for keeping score. He began playing with the blocks for incrementing variables, then reached out and shook the researcher's hand, saying "Thank you, thank you, thank you." The researcher wondered: How many eighth-grade algebra teachers get thanked by their students for teaching them about variables?

More Social

Our development of the Scratch programming language has been tightly coupled with development of the Scratch Website and online community (***Figure 4***). For Scratch to succeed, we feel the language needs to be linked to a community where people can support, collaborate, and critique one another and build on one another's work.

The concept of sharing is built directly into the Scratch user interface, making it easy for people to upload their projects to the Scratch Website, just as people share their videos on YouTube. Once a project is on the Website, anyone can run it in a browser, comment on it, add it to their list of "favorites," view the underlying program, and even revise and remix the project. (All projects shared on the site are covered by Creative Commons license.)

Since the launch of Scratch in 2007, more than 2.4 million projects have been shared on the Scratch Website. For many

Figure 4. Scratch Website and online community.

Scratchers, the opportunity to put their projects in front of a large audience—and receive feedback and advice from other Scratchers—is strong motivation. The large library of projects on the Website also serves as inspiration. By exploring projects there, Scratchers get ideas for new projects and learn new programming techniques. Marvin Minsky once noted that Logo had a great grammar but not much literature. Whereas young writers are often inspired by reading great works of literature, there was no analogous library of great Logo projects to inspire young programmers. The Scratch Website is the beginning of a "literature" for Scratch.

The Scratch Website is fertile ground for collaboration. Community members are constantly borrowing, adapting, and building on one another's ideas, images, and programs. More than 30% of the projects on the site are remixes of other projects on the site. For example, there are dozens of versions of the game Tetris, as Scratchers continue to add new features and try to improve gameplay. There are also dozens of dress-up doll projects, petitions, and contests, all adapted from previous Scratch projects.

To encourage international sharing and collaboration, we've placed a high priority on translating Scratch into multiple languages. We created an infrastructure that allows the Scratch programming blocks to be translated into any language with any character set. A global network of volunteers has provided translations for more than 40 languages.

Children around the world now share Scratch projects with one another, each viewing the Scratch programming blocks in their own language.

We have also created a separate online community, called ScratchEd (*http://scratched.media.mit.edu*), specifically for educators who are helping others learn Scratch. On the ScratchEd Website, educators share their ideas, stories, and lesson plans, learning from one another's experiences.

Future Directions

In the five years since its launch, Scratch has emerged as the most popular way for children and teens to learn to program. More than a million people have registered for accounts on the Scratch Website. But we still have a long way to go to realize Papert's dream of a fully fluent population. We are continuing to refine and extend our Scratch technology to engage a broader and more diverse audience. This summer, we will release a new generation of Scratch, called Scratch 2.0, which will enable people to program Scratch projects directly in the Web browser, providing a more seamless experience for sharing and remixing projects—and new opportunities for creativity and collaboration.

Probably the biggest challenges for Scratch—and for realizing for Papert's dream—are not technological but cultural and educational. There needs to be a shift in how people think about programming, and how they think about computers in general. We need to expand the conception of digital fluency to include designing and creating, not just browsing and interacting.

A few years ago, I gave a keynote presentation at a major educational technology conference. After my presentation, in the Q&A session, someone asked: "Wasn't Seymour Papert trying to do the same things 20 years ago?" The comment was meant as a critique; I took it as a compliment. I answered simply: "Yes." For me, Seymour's ideas remain as important today as when he published his first article about Logo in this magazine in 1971. His ideas continue to provide a vision and a direction for my research. I will be happy and proud to spend the rest of my life trying to turn Seymour's dreams into a reality. □

Acknowledgments. Many members of the Lifelong Kindergarten research group at the MIT Media Lab contributed to the ideas and technologies discussed in this essay.

Send Us Your Comments

All readers of ***Educational Technology*** *are welcome to send in comments for possible publication in these pages. Your views may deal with your reactions to articles or columns published in the magazine, or with any topic of general interest within the larger educational technology community.*

Send your Reader Comments to us at ***edtecpubs@aol.com*** *. In general, your message should be up to 750 words in length, though longer contributions will be considered, depending on the specific topic being addressed. Join in the ongoing conversation in the pages of this magazine.*