# Lifelong Kindergarten:
# Cultivating Creativity through Projects, Passion, Peers, and Play

Mitchel Resnick, MIT Media Lab
Published by MIT Press (2017)

*Excerpt from Chapter 2: Projects*


## Makers of Things

In January 2009, in a large lecture hall on the MIT campus, I watched Barack Obama inaugurated as the 44th president of the United States. The room was packed with more than 500 people, and a video of Obama's inauguration address was projected on two large screens at the front of the room. Given that the audience was full of MIT scientists and engineers, you won't be surprised to hear that the strongest reaction came when Obama declared: "We'll restore science to its rightful place." The room erupted with applause.

But that's not the line in the inaugural address that captured my attention. For me, the most memorable moment was when Obama said: "It has been the risk-takers, the doers, the makers of things—some celebrated, but more often men and women obscure in their labor—who have carried us up the long, rugged path towards prosperity and freedom."

*Risk-takers*. *Doers*. *Makers of things*. These are the X students, the creative thinkers. They've been the driving force for economic, technological, political, and cultural change throughout history. Today, everyone needs to be a risk-taker, a doer, a maker of things—not necessarily to bend the arc of history, but to bend the arcs of their own lives.

By using the phrase *makers of things*, Obama was making an implicit reference to a movement that was just starting to spread through our culture: the Maker Movement. It started as a grassroots movement, in basements, garages, and community centers, among people who had a passion for making things—and sharing their ideas and creations with one another. In 2005, the movement gained momentum when Dale Dougherty launched *Make:* magazine, celebrating the joys of building, creating, and inventing things. The magazine aimed to democratize making, showing how everyone can get involved in do-it-yourself activities. The first issue described "amazing things that ordinary people are making in their garages and backyards," providing instructions for making a kite to take aerial photographs, a thermoelectric keg wrap to keep beer cold, and glow sticks to make dynamic light patterns at night.

The following year, in 2006, Dale organized the first Maker Faire, described as a "family-friendly festival of invention, creativity, and resourcefulness." There were exhibits and workshops for making jewelry, making furniture, making robots—making almost anything you could imagine. Over the past decade, hundreds of Maker Faires have sprung up around the world, attracting millions of engineers, artists, designers, entrepreneurs, educators, parents, and children.

For many people, the appeal of the Maker Movement is in the technology. There has been a proliferation of new technologies, such as 3-D printers and laser cutters, that enable people to design, produce, and customize physical objects. Many people are excited about the business

potential of these technologies, predicting that the Maker Movement will spark a new industrial revolution, in which small businesses (or even individuals) can manufacture products that previously required large factories with economies of scale.

I'm attracted to the Maker Movement for different reasons. I believe it has the potential to be not just a technological and economic movement but also a learning movement, providing new ways for people to engage in creative learning experiences. As people make and create, they have opportunities to develop as creative thinkers. After all, *create* is at the root of *creativity*.

Perhaps most important, the Maker Movement encourages people to work on projects—the first of the four P's of creative learning. The articles in *Make:* magazine and the exhibits at Maker Faire don't just teach the techniques of making; they support a project-based approach to learning, in which people learn new ideas, skills, and strategies while working on personally meaningful projects. Dale Dougherty refers to projects as "the basic units of making."

I experienced the power of projects in a personal way as I was growing up. As a child, I enjoyed playing all types of sports: baseball, basketball, tennis, and more. But even more than playing sports, I enjoyed "making" sports. I was constantly inventing new sports to play with my brother and my cousin. I was fortunate to have a backyard for building and playing—and fortunate to have parents who allowed me to turn the backyard into a workspace for my projects.

One summer, I dug up the backyard to create my own miniature golf course. It was a continual learning experience. I started by digging simple holes in the ground for the golf holes, but I found that the holes lost their shape over time, so I began embedding aluminum cans in the holes. That worked fine until it rained, and the cans filled up with water that was difficult to get rid of. My solution: cut off both ends of the cans before embedding them in the ground so that water could drain out of the bottom.

As I added walls and obstacles on the mini-golf course, I needed to figure out how the ball would ricochet off of them. That provided me with a motivating context for learning the physics of collisions. I spent hours calculating and measuring angles so that I could bounce a golf ball off obstacles and into the hole. That experience was more memorable than any science lesson I had in the classroom.

Along the way, I began to develop an understanding about not only the process for making a miniature golf course, but the general process for making anything: how to start with an initial idea, develop preliminary plans, create a first version, try it out, ask other people to try it out, revise plans based on what happens—and keep doing that, over and over. By working on my project, I was gaining experience with the Creative Learning Spiral.

Through these types of projects, I began to see myself as someone who could make and create things. I started to look at things in the world in a new way, wondering how they were made. How is a golf ball made, or a golf club? I started to wonder what other things I could make.

If you search on the *Make:* website today (makezine.com), you'll find lots of articles describing miniature golf projects, with titles like "DIY Tabletop Mini Golf" and "Urban Putt: Miniature Golf 2.0." The technologies have evolved since I built my miniature golf course nearly 50 years ago. It's now possible to produce custom-designed obstacles with a 3-D printer or laser

cutter, and it's now possible to embed sensors in the obstacles, triggering motors or LEDs to turn on as the golf ball careens off an obstacle.

I'm still proud of the "old-fashioned" miniature golf course that I built as a child. But I'm also excited that new technologies can expand the types of projects that children can create—and inspire more children to become makers of things.

## Learning-through-Making

Over the years, many educators and researchers have advocated *learning-by-doing*, arguing that people learn best when they are actively involved in *doing* things, learning through hands-on activities.

But in the culture of the Maker Movement, it's not enough to *do* something: You need to *make* something. According to the maker ethic, the most valuable learning experiences come when you're actively engaged in designing, building, or creating something—when you're *learning-through-making*.

If you want to get a better understanding of the connections between making and learning, and how to support learning-through-making, there is no better place to look than the work of Seymour Papert. I was lucky enough to work with Seymour for many years at MIT. More than anyone else, Seymour developed the intellectual foundations for learning-through-making, along with compelling technologies and strategies for supporting it. Indeed, Seymour should be considered the patron saint of the Maker Movement.

Seymour loved learning in all of its dimensions: understanding it, supporting it, doing it. After earning a PhD in mathematics from Cambridge University in 1959, Seymour moved to Geneva to work with the great Swiss psychologist Jean Piaget. Through careful observation and interviews with thousands of children, Piaget found that children actively construct knowledge through their everyday interactions with people and objects in the world. Knowledge isn't poured into children, like water into a vase. Instead, children are constantly creating, revising, and testing their own theories about the world as they play with their toys and friends. According to Piaget's *constructivist* theory of learning, children are active builders of knowledge, not passive recipients. Children don't *get* ideas, they *make* ideas.

In the early 1960s, Seymour moved from Geneva, Switzerland, to Cambridge, Massachusetts, to take a faculty position at MIT. In doing so, Seymour was moving from the epicenter of a revolution in child development to the epicenter of a revolution in computing technology—and he spent the following decades making connections between the two revolutions. When Seymour arrived at MIT, computers still cost hundreds of thousands of dollars or more, and they were used only at large companies, government agencies, and universities. But Seymour foresaw that computers would eventually become accessible for everyone, even children, and he had a vision for how computing could transform the ways children learn and play.

Seymour soon emerged as a leader in a spirited intellectual battle over how to introduce computers in education. Most researchers and educators adopted a *computer-aided instruction*

approach, in which computers played the role of teachers: delivering information and instruction to students, conducting quizzes to measure what the students had learned, then adapting subsequent instruction based on student responses to the quiz questions.

Seymour had a radically different vision. For Seymour, computers were not a replacement for the teacher but a new medium of expression, a new tool for making things. In 1971, still five years before the first personal computer was introduced, Seymour coauthored (with Cynthia Solomon) an article titled “Twenty Things to Do with a Computer.” The article described how children could use computers to draw pictures, create games, control robots, compose music, and many other creative activities.

Seymour’s approach built on what he had learned from Piaget, viewing children as active constructors, not passive recipients, of knowledge. Seymour went a step further, arguing that children construct knowledge most effectively when they are actively involved in constructing things in the world—that is, when they are makers of things. Seymour called his approach *constructionism*, because it brings together two types of construction: As children construct things in the world, they construct new ideas in their heads, which motivates them to construct new things in the world, and on and on, in a never-ending spiral of learning.

To bring these ideas to life, Seymour and his colleagues developed a computer programming language for children, called Logo. Until then, programming had been viewed as a specialized activity, accessible only to people with advanced mathematical backgrounds. But Seymour saw programming as a universal language for making things on the computer, and he argued that everyone should learn to program.

In his book *Mindstorms*, Seymour contrasted the computer-aided instruction approach, in which “the computer is being used to program the child,” with his own approach, in which “the child programs the computer.” In the process of learning to program, he wrote, a child “both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building.”

When Logo was first developed, children used it primarily for controlling the motions of a robotic “turtle” (so named because it used a hemispherical shell to protect its electronics). As personal computers became available in the late 1970s, children used Logo to draw pictures on the screen, typing commands like “forward 100” and “right 60” to tell the “screen turtle” how to move, turn, and draw. As children wrote Logo programs, they learned mathematical ideas in a meaningful and motivating way, in the context of working on projects they cared about.

Through the 1980s, thousands of schools taught millions of students how to program in Logo, but the initial enthusiasm didn’t last. Many teachers and students had difficulty learning to program in Logo, because the language was full of non-intuitive syntax and punctuation. To make matters worse, Logo was often introduced through activities that didn’t sustain the interest of either teachers or students. Many classrooms taught Logo as an end unto itself, not as a way for students to express themselves and to explore what Seymour called “powerful ideas.” Before long, most schools shifted to other uses for computers. They began to see computers as tools for delivering and accessing information, not for making and creating as Seymour had imagined.

Seymour's ideas about learning-through-making are now starting to gain traction once again, as evidenced by the rise of the Maker Movement. Although Seymour's work on Logo began more than 50 years ago and his landmark book *Mindstorms* was published back in 1980, his core ideas are as important and pertinent today as ever before.

## Fluency

In the past few years, there has been a surge of interest in learning computer programming—or coding, as it's popularly called these days. There are now thousands of apps, websites, and workshops to help kids learn to code. Our Scratch programming software is part of this trend—but with a distinct difference.

Most introductions to coding are based on *puzzles*. Kids are asked to create a program to move a virtual character past some obstacles to reach a goal. For example, move the *Star Wars* robot BB-8 to pick up scrap metal without running into the bandit, or program R2-D2 to get a message to the rebel pilots. As kids create programs to solve these puzzles, they learn basic coding skills and computer science concepts.

With Scratch, we focus on *projects* instead of puzzles. When we introduce kids to Scratch, we encourage them to create their own interactive stories, games, and animations. They start with ideas and turn them into projects that they can share with other people.

Why focus on projects? We see coding as a form of fluency and expression, much like writing. When you learn to write, it's not enough to learn spelling, grammar, and punctuation. It's important to learn to tell stories and communicate your ideas. The same is true for coding. Puzzles might be fine for learning the basic grammar and punctuation of coding, but they won't help you learn to express yourself. Imagine trying to learn to write just by working on crossword puzzles. It could improve your spelling and vocabulary, and it could be fun, but would you become a good writer, able to tell stories and express your ideas fluently? I don't think so. A project-based approach is the best path to fluency, whether for writing or coding.

Even though most people don't grow up to become professional journalists or novelists, it's important for everyone to learn to write. So too with coding—and for similar reasons. Most people won't grow up to become professional programmers or computer scientists, but learning to code fluently is valuable for everyone. Becoming fluent, whether with writing or coding, helps you to *develop your thinking*, *develop your voice*, and *develop your identity*.

### *Developing Your Thinking*

In the process of writing, you learn to organize, refine, and reflect on your ideas. As you become a better writer, you become a better thinker.

As you learn to code, you also become a better thinker. For example, you learn how to break complex problems into simpler parts. You learn how to identify problems and debug them. You learn how to iteratively refine and improve designs over time. Computer scientist Jeannette Wing has popularized the term *computational thinking* to refer to these types of strategies.

Once you learn these computational-thinking strategies, they can be useful in all types of problem-solving and design activities, not just in coding and computer science. By learning to

debug computer programs, you'll be better prepared to figure out what went wrong when a recipe doesn't work out in the kitchen or when you get lost following someone's directions.

Solving puzzles can be helpful in developing some of these computational-thinking skills, but creating your own projects takes you further, helping you develop your voice and develop your identity.

### *Developing Your Voice*

Both writing and coding are forms of expression, ways to communicate your ideas with others. When you learn to write, for example, you can send a birthday message to a friend, submit an op-ed piece to your local newspaper, or record your personal feelings in a diary.

I see coding as an extension of writing, enabling you to "write" new types of things—interactive stories, games, animations, and simulations. Let me give an example from the Scratch online community. A few years ago, on the day before Mother's Day, I decided to use Scratch to make an interactive Mother's Day card for my mom. Before starting, I checked to see if anyone else had made Mother's Day cards in Scratch. I typed "Mother's Day" in the search box, and I was delighted to see dozens and dozens of projects—many of them created in the previous 24 hours by procrastinators like me!

For example, one of the projects started with the words "HAPPY MOM DAY" drawn on top of a large red heart. Each of the 11 letters was interactive, transforming to a word when touched by the mouse cursor. As I moved the cursor across the screen, touching each letter, a special 11-word Mother's Day message was revealed: "I love you and care for you. Happy Mother's Day mom."

The creator of this project was clearly developing her voice with Scratch—learning to express herself in new ways and integrating coding into the flow of her everyday life. In the future, I believe it will become as natural for young people to express themselves through coding as it is through writing.

(By the way, I didn't end up making a Mother's Day card for my mom. Instead, I sent her links to a dozen Mother's Day projects that I found on the Scratch website. My mom, a lifelong educator, responded with the following message: "Mitch, enjoyed viewing all the kids' Scratch cards so much . . . and I love that I'm the mother of a son who helped give kids the tools to celebrate this way!!!!")

### *Developing Your Identity*

When people learn to write, they begin to see themselves differently—and to see their role in society differently. The Brazilian educator-philosopher Paulo Freire led literacy campaigns in poor communities not simply to help people get jobs, but also to help people learn that "they can make and remake themselves" (as he wrote in *Pedagogy of Indignation*).

I see the same potential for coding. In today's society, digital technologies are a symbol of possibility and progress. When children learn to use digital technologies to express themselves and share their ideas through coding, they begin to see themselves in new ways. They begin to see the possibility for contributing actively to society. They begin to see themselves as part of the future.

As we've introduced Scratch to young people, I've been excited by what they've created—and what they've learned in the process. But what excites me most is the way that many Scratchers start to see themselves as creators, developing confidence and pride in their ability to create things and express themselves fluently with new technologies.