Computing in the Classroom

From the "Teaching Machine" to the promise of twenty-first-century learning technology

by Sophia Nguyen

On November 11, 1953, psychology professor B.F. Skinner sat in a fourth-grade math class, perturbed. It was Parents Day at his daughter Deborah's school. The lesson seemed grossly inefficient: students proceeded through the material in lock-stop, at the same pace; their graded assignments were returned to them sluggishly.

A leading proponent of what he called "radical behaviorism," Skinner had devoted his career to studying feedback. He denied the existence of free will and dismissed inner mental states as explanations for outward action. Instead, he focused on the environment and the organism's response. He had trained rats to push levers and pigeons to play Ping-Pong. A signed photo of Ivan Pavlov presided over his study in Cambridge. Turning his attention to a particular subset of the human animal—the schoolchild—Skinner invented his Teaching Machine.

Roughly the size and shape of a typewriter, the machine allowed a student to progress independently through a curriculum, answering test items and getting instant feedback with a few pulls of a lever. "The student quickly learns to be right. His work is pleasurable. He does not have to force himself to study," Skinner claimed. "A classroom in which machines are being used is usually the scene of intense concentration." With hardly any hindrance from peers or teachers, thousands of students could receive knowledge directly from a single textbook writer. He told *The Harvard Crimson*, "There is no reason why the school room should be any less mechanized than the kitchen."

Sixty years later, Skinner's reductionist ideas about teaching and learning continue to haunt public education—especially as it's once again being called upon to embrace technology. In December 2014, as part of a nationwide event promoting computer-science education called Hour of Code, Barack Obama hunched over a laptop alongside a group of New Jersey middle-schoolers, becoming the first president to write a line of code. The public-policy world frames computer science in K-12 education as a matter of economic urgency. Digital fluency is often called a twenty-first-century skill, equally necessary for personal workplace success and for the maintenance of America's competitive edge.

Teaching machines with capabilities beyond Skinner's imagining have proliferated in this century. The barest twitch of curiosity can be satisfied with swift thumbs and a pocket-sized interface. The possibilities seem endless: virtual realities that immerse

Images courtesy of Tina Groetzer

students in remote habitats or historical eras; learners mastering skills and content through digital games; kids everywhere achieving basic fluency in code, as their forebears once had to learn cursive. Even as researchers invent new ways to use machines for learning, they realize that the culture of the classroom may itself need to advance, in tandem with technology—a difficult proposition, when bandwidth is already taken up by battles over high-stakes testing, budgets, and teacher tenure.

Rich Halverson, education professor and associate director of the University of Wisconsin's Games Learning Society, diagnoses the problem this way: "When you manage an education system that's as rich in potential as ours with a sense of crisis, all crisis does is shut down possibility. We try to reach for the proven, for the stuff that works. Practices on the edge get ignored."

Our needs have changed, and our capabilities have grown—but we still speak like latter-day Skinners. The education writer Audrey Watters, guest lecturing last semester for the Harvard Graduate School of Education (HGSE) class "The Future of Learning at Scale," traced the behaviorist echoes in the current chatter surrounding tech-culture innovations like "gamification." As Skinner once used food pellets to induce pigeons to roll a ball back and forth with their beaks, corporate and education leaders alike have embraced the idea of dispensing nuggets of fun to shape desired behavior in humans. For businesses, gamification-based training promises to maximize profits and employee productivity; for schools, it seems like a way to motivate students to perform rote memorization—and to do so cost-effectively. The education system continues to pursue Skinner's goal of efficiency and automation.

Graduate School of Education professors Tina Groetzer and Christopher Dede have designed virtual realities, such as the pond opposite, for students to explore as scientists: for example, by collecting data along the shoreline or underwater, via submarine.

"The current system is outmoded," declares Paul Reville, Keppel professor of practice of educational policy and administration, who spoke at a September HGSE event. "We have a batch-processing, mass-production model of education that served us very well if we wanted to achieve a society in which we were sending a lot of people into low-skill, low-knowledge jobs," he says. "But for high-skill, high-knowledge jobs in a postindustrial information age, we need a very different system."

The digital society and economy, saturated by screens, require rethinking what school can and should do for today's schoolchildren.

Beyond Playing at Games

Meaningful choices—and the open-ended, experimental spirit of play—are essential to a deep game experience, says Jessica Hammer '99, a Carnegie Mellon professor who teaches game design. "Or else you're making what I like to call 'kick the puppy' games," she says. "Would you like to kick this puppy, yes or no? That's not much of a game." Games have more richness than is dreamt of in gamification's philosophy.

In the foundational studies of computer games conducted in the 1980s, education researchers asked what made the repetitive tasks of feeding quarters into a machine and controlling a joystick more appealing than the repetitive tasks of schoolwork. They wanted to apply the mechanics of games' reward systems to make educational software just as engrossing. Some titles became classics, many of them simulations like Sid Meier's *Civilization*, or *Oregon*

Photograph by Stu Rosner

Trail, created by three student teachers at Carleton College. Many others flopped. Their use of a drill-and-practice mechanic to teach content made them look an awful lot like multiple-choice tests, except that they used a cartoon monster to gobble up the right answer. This brittle sort of fun became known as "chocolate-covered broccoli"—the Teaching Machine, with shinier levers.

"Learning by doing has more conditions for success than teaching by telling," says Wirth professor in learning technologies Christopher Dede. This tenet has guided his decades of work in developing virtual and augmented realities for science students. His recent collaborator, associate professor of education Tina Grotzer, took more time to warm to simulations. With her background in cognitive science, Grotzer eventually came to appreciate the pedagogical value of virtual worlds, due to her particular interest in how learners reason about complex causality.

Environmental science poses a particular challenge for science teachers, she explains. Demonstrating a chemical reaction or physics principle right before students' eyes is eminently doable: a beaker fizzes; a catapult flings a tennis ball. Cause and effect occur within a graspable timeframe. Students can complete a hands-on lab activity from start to finish within a single class period.

Not so in environmental science, in which developments unfold on a much longer scale, exacerbating the mind's natural tendency to focus on events rather than processes. It's hard for students to track complex causality when it's "bottom-up, and distributed," says Grotzer. Nonobvious variables further frustrate the efforts of children (and many adults) to understand systems such as food webs and global weather patterns.

"A lot of students have been taught to see science as facts rather than a process of making meaning," explains Dede. He and Grotzer designed EcoMUVE (Multi-User Virtual Environment) as a simulation with a mystery at its heart. Students explore the environment around a pond at many different time points. They use a virtual net to trawl for organisms in the water, and other tools to record data about oxygen levels and temperature. They can plot changes over time on a graph, and use their avatar to speak with the characters strolling in the area. One day in late summer, pixelated carcasses turn up on the shore: a fish kill. The students must investigate what went wrong, proposing hypotheses and gathering evidence.

Called upon to put their knowledge to use, learners enter a different mindset than is usual in the classroom. Unlike in an exam, says Grotzer, "They don't know what information to bring to bear to the experience. They're not cued."

"For me, it's about getting them to wear the shoes of a scientist to see how they fit," says Dede. "The primary barrier is that they don't think they can do it."

He doesn't characterize his virtual environments as games, though they bear a family resemblance. They often have objectives (find out why the whale has washed up on the beach; trace an

"Coding for All"

With schools more eager to welcome coding in the classroom, some advocates now push to make it a public-education priority. In her 2014 book *Connected Code: Why Children Need to Learn Programming,* Yasmin Kafai, Ed.D. '93, of the University of Pennsylvania, urges schools add on to the traditional "3 Rs" of reading, writing, and arithmetic: the aRts and pRogramming. That the public perceives computers as both essential, and essentially opaque, is a form of illiteracy. Jane Margolis, Ed.D. '90, senior researcher at UCLA's Graduate School of Education and Information Studies, argues that this "learned helplessness" has larger implications for equality.

Margolis's book *Stuck in the Shallow End* continues to be one of the few lengthy examinations of how an early section of the pipeline—public K-12 education—creates racial disparities in the field of computer science. Skeptics have dismissed the "coding for all" movement as a faddish boutique reform, myopically market-driven even as it claims to advance children's problem-solving skills. But as technological innovations drive virtually every industry and shape social spaces online, advocates like Margolis view computational participation as central to the health of democracy. "Computer science can help interrupt the cycle of inequality that has determined who has access to this type of high-status knowledge in our schools," Margolis and Kafai wrote in *The Washington Post* last October. "Students who have this knowledge have a jump-start in access to these careers, and they have insight into the nature of innovation that is changing how we communicate, learn, recreate, and conduct democracy."

Despite the free programming resources available online for learners who know where to look, cultural barriers remain. Perceptions abetted by the hagiography of figures like Mark Zuckerberg '06 and Bill Gates '77, LL.D. '07, inhibit wider participation: that the path to prowess swerves away from institutions like school, and that some individuals naturally gravitate to computer science because they are innately talented and freakishly autodidactic. In reality, the typical boy genius has a great deal of what Margolis calls "preparatory privilege"—if not tech-savvy parents and summer-camp enrichment, then usually a peer group logging on together after school.

Noel Kuriakos, a member of the online educators' community ScratchEd, is a math and science teaching fellow at the tuition-free Mother Caroline Academy, a majority black and Latina girls' school in Boston. His experience in a community where many households still don't have Internet access has taught him that extracurricular outlets won't suffice: "This is where schools can play a huge, huge role, and make a big difference—in saying, 'Well, maybe not in your home, and maybe not in your social circle, but *in school* you can have access. You can do this.'"

Meanwhile, atop the structural issues that, in less affluent districts, impede learning in all subjects—underfunding, overcrowding, teacher attrition—computer-science education suffers from a special neglect affecting public schools across income ranges. Historically, it has been lumped in with home-economics class—"which, as you can imagine, makes it an attractive proposition to many computer scientists," Kafai observes wryly. Most states lack curricular standards or a teacher-certification pathway in the subject. As a result, Margolis says, schools end up "tech-rich, but curriculum-poor."

epidemic through a nineteenth-century town), but because that objective is discovery, the activities have a noncompetitive, exploratory bent. Calling it a "simulation" rather than a "game" also lowers schools' resistance to trying out the new activity. After three decades of experience, Dede firmly believes that "psychological and cultural barriers seem to slow education down more than every other field."

One day in late summer, pixelated carcasses turn up on the shore: a fish kill. Students investigate what went wrong.

Even so, "We've loosened up about games," says Eric Klopfer, director of MIT's Education Arcade, who has also worked with Dede on augmented realities. Klopfer recalls a time when teachers would tell him, "'Just don't use the word *game* in my school, because the principal will kick it right out.' And now, in fact, there are people who are saying just the opposite: 'Ooh, is that a game? I'd love to try that out in my school.'"

With the founding of hubs like Wisconsin's Games Learning Society (GLS) and MIT's Games to Teach Initiative (the forerunner to the Education Arcade) in the early 2000s, the ventures of the more risk-tolerant academic world have fed the larger industry new pedagogical models, game projects, and the occasional young talent. In turn, without investing in the educational market themselves, the biggest companies support the diversity of the larger habitat. Zynga (behind *FarmVille* and *Words with Friends*) operates co.lab, offering office space, tools, and mentoring to young ed-tech companies, including some that germinated as university projects. Electronic Arts (*SimCity, Madden NFL*) funds the nonprofit GlassLab, which develops its own games and aspires to be a resource for commercial developers who need assessment metrics and data to make their projects more educationally sound.

"Have you been on iTunes lately?" says Rich Halverson. "Good God, the world of games and games for learning is at an unprecedented glut!" But, he notes, demand has lagged. Education games remain marginal in schools: a special treat for kids who finish their work, or a remedial intervention for those who can't. Halverson believes that this underuse of a powerful resource further widens the digital divide already disadvantaging poor and minority students. Families who know of and can afford these enrichment channels will seek them out.

Halverson cites the linguist James Paul Gee, whose 2003 book *What Video Games Have to Teach Us About Learning and Literacy* claimed that *Pokemon* could be considered "perhaps the best literacy curriculum ever conceived." Its trading cards enabled millions of kids to master a complex taxonomy of imaginary creatures, all with specialized traits. Compare the average middle-school classroom to what Halverson calls the "learning space" within games themselves, and in the culture surrounding games—for example, the online game *Minecraft* and its array of blueprints, discussion boards, and how-to videos. Which provides a more authentic model for how to pursue work and personal interests in the twenty-first century?

"Think about how you do your work," Halverson instructs. "You're probably sitting in front of a computer right now. You've got a big project you're trying to come up with. You're on the phone talking to some dude from Wisconsin, taking notes. You probably have something on your wall with all the sources you're going to put together for the article. There are online resources and how-to guides for how to write. You're putting all of this stuff together. You've created your own learning environment."

From Playing to Programming

"GAMES are perhaps the first designed interactive systems our species invented," writes Eric Zimmerman, a games designer and professor at New York University. "Games like Chess, Go, and Parcheesi are much like digital computers, machines for creating and storing numerical states. In this sense, computers didn't create games; games created computers." In his essay "Manifesto for the Ludic Century," Zimmerman argues that the rise of computers parallels the resurgent cultural interest in games. Future generations will understand their world in terms of games and systems, and will respond to it as players and designers—navigating, manipulating, and improving upon them.

Yasmin Kafai, Ed.D. '93, an education professor at University of Pennsylvania, first explored how game creation and computer programming could be brought together in the classroom while

Margolis has helped write a high-school curriculum, "Exploring Computer Science" (ECS), that intends to expose students to a broad range of topics, including HTML website design, data analysis, robotics, and programming through Scratch. This will be paired with a professional-development course for teachers, who will learn inquiry-based pedagogy along with the content itself. ECS has received the backing of Code.org (the nonprofit behind the nationwide awareness event, Hour of Code), and has been adopted by districts in Los Angeles, Spokane, Chicago, and New York City, among others.

Now, Kafai is collaborating with Margolis to create an electronic textiles unit for ECS. By bringing together the crafts of circuitry design and sewing, they aim to appeal to girls by casting code as malleable, and engineering as an aesthetic pursuit. Because curriculum design in computer science remains largely uncharted territory, Kafai believes that researchers should keep the options open. "For computation, you need different materials and activities—it can't all be robotics, it can't all be game design. We need a whole array, to tease out what activities are good for which concepts, and which age levels. Because honestly, we don't really have anything right now."

Assistant professor of education Karen Brennan proposes that integrating computational thinking into classes could follow the language arts model, in which a class is devoted to the craft of manipulating words, but text, of course, is used in the service of other subject areas as well. With code too, "there's specialization, but there's also a role for it in everything you're trying to do," she explains. Though Brennan cautions that "Scratch doesn't solve every problem," she also adds, "If you start with the learning, you will not be led astray."

With Scratch (opposite), students can code interactive stories, games, and animations. With ScratchEd, HGSE professor Karen Brennan and Michelle Chung, Ed.M. '10, are building a community of educators who support each other's efforts to bring programming into more classrooms.

a Harvard graduate student working in the MIT lab of Seymour Papert. As early as the 1960s, when salesmen still hawked versions of the Teaching Machine from door to door, Papert pioneered the idea of computers in the classroom, paired with a radically different philosophy: "constructionism." This theory proposed that learners do not passively receive knowledge but actively build it—and that they do this best when they get to manipulate materials in a way that feels meaningful.

For "Project Headlight," a program beginning in 1985 that brought hundreds of computers into a public elementary school in Boston, Papert's team developed activities that immersed students in the programming language he had invented, LOGO. Following his pedagogical ideals, the researchers didn't want the computer lab to seem like a locked room at the end of a long hallway, essentially removed from daily life and learning. They wanted to know what students were naturally interested in. "These were the days of *Sonic the Hedgehog, Super-Mario*," recalls Kafai. "Kids told me that they really wanted to program their own games."

For a brief period in the early 1980s, it was relatively commonplace for avid gamers to dabble in programming; a range of books taught users how to make or modify games using languages like BASIC and Pascal. Kafai designed a curriculum in which older students would design software for children in the lower grades, in subjects ranging from fractions to marine habitats. The kids quickly realized that creating Nintendo-style games was beyond their skill set, but in art class Kafai had them think like professionals, designing boxes and creating advertisements of the kind that they might find in stores.

Ironically, the advent of multimedia CD-ROMS and software packages—and soon after, Internet browsers—made the personal computer feel so friendly that programming seemed irrelevant to its operation. Now that computers came pre-loaded and densely written with default systems and applications, they did everything the average user required. This technological wizardry deterred people from peeking behind the curtain. Creation and design were once again thought of as the province of experts; schools restricted themselves to teaching PowerPoint and touch-typing.

As programming was exiled from the classroom, researchers like MIT's Mitchel Resnick and Natalie Rusk, Ed.M. '89, gave it safe haven in the extracurricular context, through a program called the Computer Clubhouse. In true constructionist style, Resnick and Rusk envisioned members learning through design activities: controlling robots, digitally composing music, editing an animation. With the support of adult mentors and teachers, Clubhouse youth would build computational confidence. Within 15 years, the network of Clubhouses had spread to more than 100 sites, with a focus on low-income communities.

Starting in 2003, Resnick collaborated with Kafai and others to develop the programming tool Scratch, imagining that it would be used in Clubhouse-style settings. In some ways, Scratch was the inheritor of LOGO's legacy, but with a few key differences. Where LOGO had been designed with mathematics in mind, Scratch was intended to be media-centric, a tool for self-expression: kids loved the idea of making their own interactive stories, animations, and games, hardly realizing that the projects made use of algebra and algorithms. Additionally, the Scratch "grammar" would be composed of command "blocks" that could snap together, like Lego bricks, to achieve different effects. This liberated learners from the frustration of typos, or the flummoxing syntax of traditional programming languages. For a finishing

Photograph by Stu Rosner

touch, the interface would have a prominent "Share" icon—and along with it, an online community where users could display, comment on, and peer into the backend of projects. The research team built social values like collaboration, tinkering, and remixing into the culture of coding itself.

Since Scratch launched in 2007, it has been translated into more than 40 languages and used by millions worldwide—including people well outside the target age group (eight to 16), like the hundreds enrolled in Harvard's most popular class, CS 50: "Introduction to Computer Science." It's even used in primary school classrooms. In 2009, the online community hosted by MIT spun off the forum ScratchEd under the leadership of Karen Brennan (now an assistant professor of education at HGSE). There, educators can show off sample lessons and offer troubleshooting and other advice. Though their local administrations may not be able to mandate computation in the classroom, in this virtual space they can show one another what's possible.

Teaching the Teachers

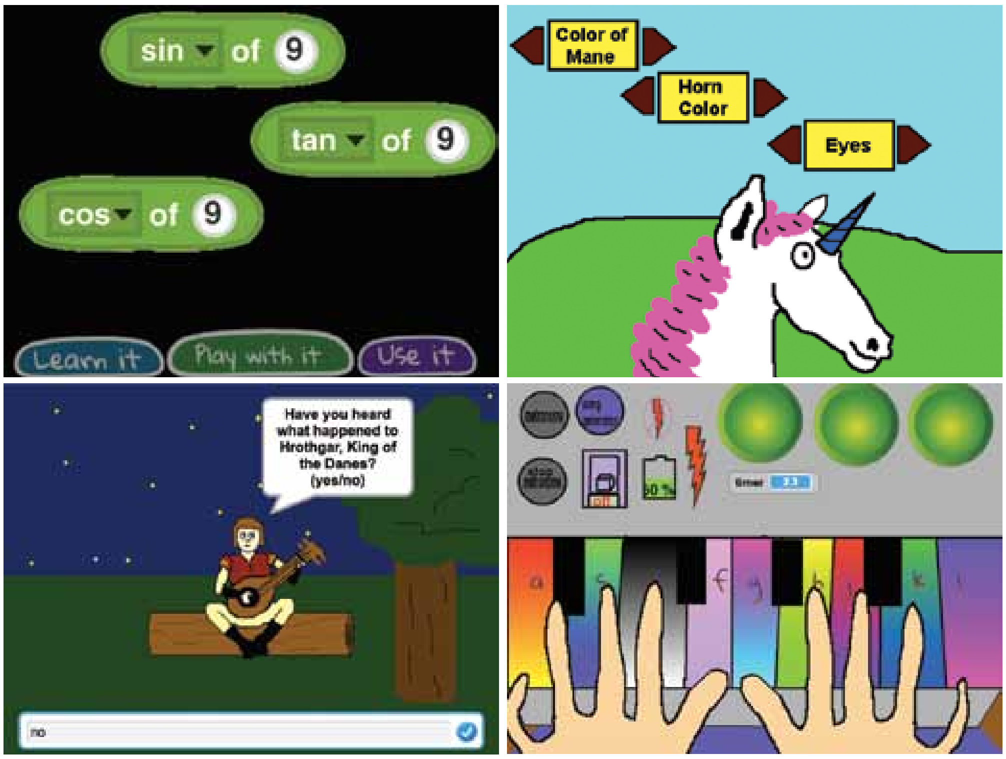
As others work toward systemic change on the policy level, urging states to create certification pathways for computer-science teachers and establish standards in the subject, Karen Brennan and the ScratchEd research team exert their efforts from the opposite end. They want to empower educators to integrate coding into the classroom independently, absent official guidance and mandates. Last fall, Brennan published a free Creative Computing Curriculum Guide that she created with former student Christan Balch, Ed.M. '14, and ScratchEd research program manager Michelle Chung, Ed.M. '10. Its 150 colorfully designed pages are divided into manageable units and sessions, under cheerful section headings that suggest "Possible Paths" and "Things to Try," and offer space for "Notes to Self" and "Feeling stuck? That's ok!" Chung says that they wanted to actively promote a "choose your own adventure" ethos, so teachers will feel emboldened to adapt the lesson plans freely. If Skinner once compared teachers to line cooks, Brennan and her team imagine them as chefs.

A growing component of their work is to convene educators in person as well as online. After the first few tutorials they ran, Brennan remembers, "We had this puzzle." Participants kept returning to introductory workshops long after they'd stopped needing what she calls "our song-and-dance 'Introduction to Scratch' piece." The program leaders soon realized that the attendees were attracted less to the workshop's content than to the opportunity to interact with colleagues. So ScratchEd began to host monthly meetups at the MIT Media Lab, welcoming participants at all levels to gather, collectively set an agenda, and share their frustrations, success stories, and expertise.

Soon the sessions attracted teachers from as far away as New York and Philadelphia—and once, a woman who'd taken a red-eye flight from San Diego to make the Saturday morning meeting. The team realized that they had hit upon an unexpected vein of potential energy. "What is that ephemeral quality? Can we package that, can we communicate it? Because it's very different from many types of professional learning that teachers are encountering," says Brennan.

In the fall of 2014, a trio of local educators took the reins of the original meetup group. They now host the gatherings at Kennedy Longfellow School in Cambridge, in a computer lab recently remodeled to accommodate round tables surrounded by carts of robotics kits, laptops, and tablets. For the November meeting, held on the cusp of the nationwide Computer Science Education Week and the Hour of Code, participants included teachers, parents, and library and technology specialists, from public and private schools. Some wanted to learn strategies for running an after-school club; others wondered how to convince other teachers to allocate already scarce time to the uncertain prospect of grappling with computers. Empathy, and experience within the system, cooled their natural evangelism; none wanted to pile onto the pressures most teachers already feel. Talk of individual projects—using Scratch to build a hurricane simulator, or to animate verbs as a study aid for Spanish and French classes—led to deeper discussions of classroom dynamics: colleagues' general reluctance to take risks, or how a coding activity often goes more smoothly when the kids take charge.

Heather French, Ed.M. '13, who has worked as an instructional technology specialist in the Cambridge public-school district since her graduation, attests that teaching "digital confidence is often the most challenging part of my job." When frustrated by the technology they use—whether a misformatted Google Application or a tangled snarl of code—children and adults reflexively



Images courtesy of Karen Brennan

ask her to intervene. A self-taught programmer herself, French believes that part of her job is to teach them how to find the answers themselves: "They just need the courage to try."

In Skinner's ideal school, every learner is an island, in front of a glowing screen; teachers are reduced to technicians.

Students are accustomed to feeling this uncertainty; teachers, less so. Implicitly, many regard expertise as their source of legitimacy: a store of knowledge, in the form of facts, to be transmitted to the children. They want all the solutions to all possible problems before they feel comfortable leading a lesson—and because computers are only beginning to return in the classroom in these new ways, few have that expertise.

In a talk called "Getting Unstuck" for an HGSE-wide event last September, Brennan addressed teachers' fears of encountering an intractable technical issue that causes the day's lesson to break down. During her research, she reported, young independent Scratch learners had suggested some strategies for when a project malfunctions: reread the code with a critical eye, to check for mistakes; enlist the help of a collaborator, to see the project with fresh eyes; find successful examples to analyze and emulate. These were helpful, Brennan said, but also suggestive of a broader lesson—that teachers could model problem-solving for their students even if they didn't have all the solutions. Indeed, a small degree of uncertainty might be preferable: making room for more spontaneous discovery, and more authentic and rewarding classroom interactions.

With its collaborative spirit, the meetup is a mode of professional development that matches this pedagogy. Recently, Brennan and Michelle Chung released a Meetup Kit in the style of their curriculum guide, so that the model can be replicated and remixed elsewhere.

Rebooting School

At the cambridge gathering last fall, there was a general sense that the rising profile of events like Hour of Code could make real inroads into schools' uncertainty about investing more energy in tech. Still, a persistent worry encroached on the excitement: that the enthusiasm would come up against calcified classroom culture and dissipate. How could they prevent digital media from becoming a faddish one-off, forgotten amid competing demands on educators? "My fear," confessed Ingrid Gustafson, one of the meetup's core leaders and one of French's technology colleagues in the Cambridge schools, "is that we're building a path to nowhere."

When she's not bothered by such doubts, Gustafson speaks glowingly about projects that, to her, seem like signs of a way forward. Last year, she helped develop an activity for sixth-graders who had recently played with Grotzer and Dede's EcoMUVE forest simulation and were then assigned to present what they hypothesized about the virtual world they'd explored. After a visiting artist taught them about scientific illustration and painting techniques, the students drew food webs as accurately and aesthetically as possible. Then they used MakeyMakeys—circuitry toolkits that make ordinary objects into touchpads—to hook the paper models up to Scratch simulations that they coded themselves, practicing the new computational concepts they had learned, like loops, conditionals, and sequencing. When a student touched a deer on the food web, it would appear in the virtual ecosystem and interact with other organisms.Beyond the lessons in art, biology, and computer science, the sixth-graders learned something deeper and possibly more enduring: that the digital realm is not just a received environment, with expertly designed features beyond their control; it's a world in which they can communicate and create. Gustafson recently received a $15,000 state grant to bring the EcoMUVE activity to all sixth-graders in the district. She and her colleagues hope that this new introduction to Scratch programming will work in tandem with the existing robotics unit for the seventh-graders. The faculty will build upon their collective knowledge, institutionalizing computational creativity one year at a time.

As the digital realm has permeated almost every aspect of modern life, institutions like schools remain vital levelers. They promote more democratic and equitable participation in society's virtual marketplaces and town halls. A public with staggeringly uneven rates of digital illiteracy creates ravines between the creators and the users; those who design the system and those at its mercy. As media theorist Douglas Rushkoff warns, "Program or be programmed."

B.F. Skinner's machine was rudimentary, its interface only the narrowest of windows through which students squinted at curricula printed on cylinders of paper. The windows are so much larger now, offering portals to seemingly infinite information. The advanced features of the new Teaching Machines could be used to realize Skinner's ideal school: every learner an island, in front of a glowing screen; all students proceeding at different paces through the same exact motions; teachers reduced to technicians.

Indeed, when computers first entered classrooms on a mass scale, it was as banks of monitors installed to speed up rote learning, under a "one size fits all" philosophy. Seymour Papert criticized this transformation as "the shift from a radically subversive instrument in the classroom to a blunted conservative instrument in the computer lab." He proposed an alternative to the concept of the computer programming the child: "In my vision, the child programs the computer." He imagined another way for machines to revolutionize classrooms.

The call to reexamine *what* teachers teach can bring renewed discussions of *how*. With tools like augmented reality, games, and coding, it's possible to imagine a model of schooling that departs from its behaviorist past—creating a Ludic Education for a Ludic Age, promoting inquiry, collaboration, experimentation, and play. In this vision, teachers and students are partners in a joint venture. They open up the Teaching Machine to peer into its guts and gears—tinkering, failing, and trying again, to see what they can make of it together. The machines can return education to what it's always been: a project that's intrinsically human.

Sophia Nguyen is a staff writer at the magazine.