Responding to the Coding Crisis: From Code Year to Computational Literacy

Kevin Brooks, North Dakota State University
Chris Lindgren, University of Minnesota

> *[T]he discourse of literacy crises engages deep-seated cultural anxieties and attempts to resolve them magically, by regulating the production and use of literacy and by drawing lines between standard English and popular vernaculars, "masters" and "servants." -- John Trimbur.*
>
> *In the emerging, highly programmed landscape ahead, you will either create the software or you will be the software. It's really that simple: Program, or be programmed. Choose the former, and you gain access to the control panel of civilization. Choose the latter, and it could be the last real choice you get to make. -- Douglas Rushkoff.*
>
> *This is a fabulous product at the right time. -- Codecademy.*

Introduction

Cultural anxiety about the negative effects of media and ubiquitous computing on traditional literacy skills has persisted into the 21st century in the form of what John Trimbur (1991) calls a narrative of literacy decline (p. 281). A second, powerful narrative of "rising literacy expectations" (Trimbur p. 281) has emerged in response to the same conditions: everyone should learn to code, or, everyone (because of this call for universal education), should develop computational literacy. This coding crisis discourse can be traced to Douglas Rushkoff's (2010) *Program or Be Programmed* in which he argues that the US is losing significant global power, and the fate of all humanity is on the line, because, and we are paraphrasing Trimbur, "the kids still can't code." "By this account [of rising expectations]" Trimbur writes, "literacy crises take place when a cultural lag occurs, when literacy practices and education have not quite caught up to increased expectations and heightened demand. . . . In this view, a heightened demand for literacy is linked to job performance, productivity, and the need to improve the competitive position of the United States in the world market" (p. 284). The coding crisis discourse has followed the pattern of a rising expectations narrative exactly as Trimbur articulated.

After Rushkoff's book appeared, crisis discourse began circulating in *The Chronicle of Higher Education* (Salter 2012) and it appeared in various guises within edited collections like *Debates in the Digital Humanities* (Gold 2012). Attempts to address and capitalize on the crisis discourse resulted in the Codecademy, an online educational company, proclaiming 2012 to be "Code Year," followed in 2013 by Code.org's "Hour of Code" (2013a) the ultimate easy on-ramp for learning this apparently essential skill. As of this writing, almost 40 million users have tried Hour of Code and 1.425 million have signed a petition to give every student in every school in America a chance to learn computer science. Code.org has enlisted the ethos of a variety of public figures to endorse coding as the key to personal success and global competitiveness, but also fuel the coding crisis discourse (see Figure 1). In 2014, coding crisis stories have been running regularly on National Public Radio (NPR Staff 2014) in Newsweek (Maney 2014) and in Time Magazine (Nicks 2014). The crisis has escaped the academy, become a "headlong global frenzy to teach programming" (Maney 2014) and seems to have overshadowed the narrative of literacy decline, for now.

Figure 1. Screen capture of Code.org's home page on Feb.12, 2013 with images and quotations from thought leaders about the value of learning how to code.

What role, if any, do literacy experts have to play in responding to the coding crisis discourse? Trimbur (1991) noted that "Writing teachers and writing-program administrators have not been immune to the rhetorical power of [a] literacy crisis," and "There is little question that writing programs and composition studies have been direct beneficiaries of the current literacy crisis" (p. 277). Sure enough, a Town Hall session at Computers and Writing 2012 took up Ruskoff's slogan: "Program or Be Programmed: Do We Need Computational Literacy in C&W" (Sample & Vee 2012). The consensus of the panelists was yes, with David Reider and Karl Stolley offering no caveats, Annette Vee, Mark Sample, Alexandria Lockett and Elizabeth Losh offering more moderate views including Vee's call to value all forms of code-writing styles and practices and Sample's suggestion that we strive for "code competency" rather than literacy. The panelists' collective message seems to be that if rhetoric and composition teachers and scholars want to participate in the full range of expressive powers available today, and if our field wants to fully engage with the capabilities of the computer as a communicative medium, we need to seriously consider how we can develop our own and our students' computational literacy.

So no brainer, right? Literacy scholars, and particularly the computers and writing subfield, have lots to gain from taking up computational literacy: status as writers and coders, expressive and perhaps institutional power, funding from NEH for digital humanities projects or even NSF for collaborative efforts in pursuit of a "Computational Education for the 21st Century" grant. If "computational literacy" is folded into literacy development efforts, we will be doing the right thing for our students, ourselves and society.

Maybe, maybe not. We have to admit that we, the authors, have been swept up by the coding crisis discourse, and formed a working group (2010-13) we called “Sugar Labs @ NDSU” that focused on learning how to effectively use the open source operating system “Sugar” and introduce K-6 students to computational literacy. Our goal has been to increase our understanding of computational thinking while also bringing computational literacy to local students. Our team of six, led by the authors, ran a pilot program in 2010-11, then a 14-week curriculum in 2011-12 with thirteen students from our city’s most diverse and economically disadvantaged elementary school. We expanded the curriculum to a second school in 2012-13, but we draw almost exclusively from the 2011-12 data in this article. We had fun, the kids had fun, they may have learned some things about computational thinking, but we might also have been playing right into a pernicious discourse that touts 21st century skills and upward mobility. We might have inadvertently sorted students into “programmers” and “programmed” along gender and race lines. We might have been privileging technical skills and a screen-obsessed culture while neglecting our discipline’s most powerful technology, language.

Trimbur tried to warn us. “In the flurry of activity it has taken to launch new courses and programs,” he says of the growth composition studies and writing programs saw in the 70s and 80s, “we have not stopped often enough to ask what we are subscribing to when we say the magic words ‘literacy [or, now, coding] crisis’” (p. 278). This essay is an opportunity for us to stop and assess 1) the coding crisis discourse generated by Rushkoff, Codecademy, Code.org, and others, and 2) our own attempts to intervene in this apparent crisis by offering access to computational literacy to fourth and fifth graders through an afterschool curriculum. Before we and others embrace this discourse fully and seek benefits from it, we also need to make sure we understand what we are gaining, losing, and inadvertently subscribing to.

In the section of our chapter entitled “What are we subscribing to?” we will examine how three concepts Trimbur found at work in the literacy crises of the 70s and 80s—global economic competitiveness, individual credentialing, and social stratification—have played out in the current coding crisis discourse. We provide some historical and local context for our efforts in “Crisis Intervention Planning” and we analyze the limitations and successes of our efforts in “Crisis Intervention Analysis.” Drawing on field notes and video data, we will acknowledge the problems and successes in our afterschool code literacy efforts, avoiding both the crisis and salvation rhetoric that dominates this and other literacy crises. In the conclusion, we support the development of computational literacy but argue that a historical and critical perspective on the crisis should modulate our rhetoric and encourage a long-term, sound approach to computational literacy rather than looking for magical solutions.

What are we subscribing to?

When Trimbur stopped to ask what educators, literacy and composition scholars had been subscribing to over a century of literary crises, he found two general narrative patterns and a number of specific values not typically identified in the crisis discourse. He found that crises were typically stories about declining literacy abilities or rising literacy expectations (p. 281), and the coding crisis discourse clearly partakes of the latter. Trimbur identifies at least three unexamined assumptions in the narrative of progress: 1) that increasing literacy standards will increase American global competitiveness; 2) that the call to increase literacy standards is primarily about increasing individuals' credentials and private interests rather than the public good; and closely related, 3) that individuals who do not attain these higher credentials fail because of personal shortcomings (pp. 284-94). Below we elaborate on what Trimbur has to say about these assumptions, and then examine contemporary coding crisis discourse in light of Trimbur’s insights.

A rhetoric of global competitiveness.

"[A] heightened demand for literacy is linked to job performance, productivity, and the need to improve the competitive position of the United States in the world market" (p. 284). Writing in the early 1990s, Trimbur acknowledges the economic transformation brought about by the "computerization of information" but he raises doubts about the need for higher literacy levels "when nine new jobs created are for cashiers and checkout clerks to each one for computer programmers" (p. 285). As Trimbur writes in the final essay of this book, the vast changes brought about by the digital revolution were difficult to foresee in 1991. Trimbur would have a more difficult time doubting the real need for highly educated workers in 2014[1], but the coding crisis narrative over-emphasizes global economic competitiveness at the expense of computational literacy's ability to empower users' expressive, aesthetic, and rhetorical abilities. Just as literacy narratives can foreground the democratic, creative, and aesthetic values of reading and writing, coding literacy narratives can avoid the discourse of crisis and foreground both individual and collective empowerment outside the standard institutions of contemporary meritocracies.

Rushkoff (2010) tells a story of declining abilities in order to make support his call for raising the literacy standards:

> That's right: America, the country that once put men on the moon, is now falling behind most developed and many developing nations in computer education. We do not teach programming in most public schools. Instead of teaching programming, most schools with computer literacy curricula teach programs. Kids learn how to use popular spreadsheet, wordprocessing, and browsing software so that they can operate effectively in the high-tech workplace. These basic skills may make them more employable for the entry-level cubicle jobs of today, but they will not help them adapt to the technologies of tomorrow. (p. 135-36)

Rushkoff is expressing the kind of deep-seated cultural anxieties Trimbur says come with literacy crises, while Codecademy, an online educational company, provides the kind of magical solution that accompanies the anxiety (Trimbur, p. 279). Their website's landing page offers three pithy solutions to the problem: get technical, learn how to hack, and use this "product" (see Figure 2).

Figure 2. Screen capture of Codecademy's splash page on Nov. 20th, 2012.

1 The March 2012 Bureau of Labor Statistics (BLS) projects 17% job growth for retail sales workers and only 7% growth for cashiers between 2010 and 2020, but BLS projects 12% growth for computer programmers, 28% for network administrators, 22% for information security analysts, web developers, and computer network architects, and 30% for software developers. Even technical writers, a career that might embrace computational literacies, in and out of code, better than any other career, is projected to grow at 17%. The shortfall of labor, particularly a diverse labor pool, is well documented in computer science and related fields, giving some credibility to the economic side of the coding crisis discourse.

Admittedly, in each of our Sugar Labs proposals for funding, we included language of global competitiveness and credentialing (the next section). We said in our Google Rise Proposal that one of the benefits of our afterschool program would be to "Establish a progressive computing culture within the Fargo community that will benefit local high-tech companies by ensuring there is a qualified pool of job candidates" (Sugar Labs @NDSU, "Google Rise Proposal"). These economic competitiveness arguments are commonplace in the narrative of heightened expectations, and Trimbur might say that crises are "always articulated in a relation to power and the negotiation of cultural hegemony" (p. 285). A better argument, and one that would resist the discourse of crisis, would emphasize the **global cooperation** that characterizes the work of the One Laptop Per Child Foundation and Sugar Labs network. Our local deployment was connected to global deployments of Sugar through the XO computer; resources and ideas were shared among that community through listservs and blog aggregation. Going forward, we will have to consider more carefully the rhetoric we deploy, resist the crisis rhetoric, and stick more closely to the principles we actually subscribe to.

Individual credentialing rather than collective action

Global economic competitiveness could be conceived of as a collective endeavor, but Trimbur notes the theme of individualization that runs through literacy crisis discourse. In his conclusion, he writes, "part of the ideological work performed by the discourse of crisis has been the privatization of literacy —the representation of reading and writing not as a means of enlarging the public sphere of discourse and political participation but as personal credentials, forms of cultural capital and articulations of a wider ideology of possessive individualism" (p. 294). Trimbur documents the rich history of "popular literacy in the early 19th century [that] belonged to the civil society" and he notes the efforts of slaves to "clandestinely . . . learn to read and write, for religious reasons and an act of political resistance" (p. 288). Literacy and computational literacy have not always, nor need always, be cast as individual cognitive skills or credentials.

In Rushkoff's book and articles like "Learn to Code, Get a Job," he gestures to the public value of learning how to think computationally, but the rhetoric of social value is still framed by the rhetoric of individual credentialing and economic or military competitiveness.

> If you know how to code, you can get a high-paying job right now, or make valuable stuff right now. You will understand more about how the world works, and become a participating member in the digital society unfolding before us. You will be enabling America to compete effectively on both the economic and military frontiers, where we are rapidly losing our competitive advantage due to our failure to teach ourselves code. We should not have to wait for the NYSE to be hacked by kids from Asia to learn this lesson.

Codecademy is not oriented to empowering activists to harness the power of codes or databases; instead, it teaches participants how to write games such as black jack, hangman, checkers, and how to code a flight scheduler program. Coding, in this manifestation, is not "a weapon to defend democratic principles, a means to curb the power of the state and the attack the elitism of existing institutions" as Trimbur describes one of its early 19th century functions (p. 288). The "kids from Asia" who hacked the NYSE might have embraced that vision, but the coding crisis in America has been framed primarily as one of credentialing for individual success and global competitiveness. Code.org embraces something closer to a socially transformative vision, but that vision is presented as a choice, the other being personal success. Code.org's (2013b) founder Hadi Partovi says in the site's promotional video, "Whether your want to make a lot of money or change the world, computer programming is an incredibly empowering skill to learn."

Our “Smart Computing Culture” proposal that generated funding for Sugar Labs @ NDSU didn’t offer formal credentialing, but we claimed we would “Expand children’s understanding of technology beyond cell phones and video games to build logic and technical skills that will enable them to succeed in a technological environment.” Our pedagogy, however, was collaborative and constructionist; we used no badges or reward system to privilege individual accomplishments. We offered no certificate of completion. Looking at our founding documents through the rhetoric of a literacy crisis, we clearly subscribed to the dominant discourses even if our practices embodied other core values.

An unarticulated social stratification

The crisis of heightened expectations in the 1980s, Trimbur suggests, is entwined with Reaganomics, the two Americas that resulted, and a culture of blame. “[L]iteracy appears both as a social explanation that individualizes oppression by blaming the victim and as a tool to incorporate all the ‘other Americas’—the poor, blacks, Hispanics, new Asian immigrants—into a monolingual body politic” (p. 286). In other words, those who cannot keep up with the literacy demands of a postindustrial society—real or imagined—have only themselves to blame, and as Americans, they have been given an unprecedented opportunity to succeed. Those who succeed apparently become fully assimilated.

This point about unarticulated social stratification is of particular importance because its values are hidden, rather than embraced or touted by the charged language of the coding crisis discourse. Rushkoff does not include “increase access” as one of his ten commandments for the digital age, nor do Codecademy’s assignments, like hangman and black jack, seem likely to broaden participation by gender, race, or socio-economic status. Efforts like ours, Google’s, and NSF’s might actually backfire if opportunities are presented, but the initiatives don’t work because organizers don’t fully understand and address the deeply embedded cultural issues and history that have lead to the computation sciences, and the digital humanities being so white (McPherson 2012). As long as computer programming is defined as an individual, cognitive skill and specific segments of the population are not programmers, the crisis discourse sets up the possibility of individual, rather than systemic failure.

In both our “Smart Computing Culture” (2011) and “Google Rise” (2012) grant applications, we identify the diversity of the students in the program, balanced by gender and race, but we show no specialized knowledge of how to make sure a diverse group will succeed. We say in our Smart Computing Culture grant that we “want to ensure that children are aware of career paths that are available and start building the necessary skill set early in life,” but we do not show an awareness of the leaky pipe in STEM disciplines, and we do not provide a plan for counteracting either the cultural values that inform popular understanding of computer programmers or the actual culture that permeates computer science departments or coding—intensive work environments. In the Google Rise proposal, we recognize that we would need to connect our students to middle school and high school computational literacy opportunities in order for our elementary school intervention to have any lasting effect. Our community is not without opportunities for sustained development of computational literacy, but without attention to issues of social stratification, the typical demographic—male, white and Asian—is likely to take the most advantage of these opportunities.

Trimbur's tropes of crisis discourse helps illuminate the very issues we hoped to address with our afterschool program, Sugar Labs @ NDSU. We knew that the digital divide is materially more than access to technology, and that culture is embedded in the design and appropriation of technology. Accordingly, our program allied itself with a computational education movement that has been churning since the 1960s. In what follows, we discuss the cultural and historical roots of the technology we used in our afterschool program, the Sugar operating system (OS). In the next section,

Crisis Intervention Planning, we discuss our association with this computing culture, and how we implemented it within an afterschool context.

Subscribing to computational literacy over 'learning to code'

In the 1960s, Seymour Papert, Daniel G. Bobrow, Wally Feurzeig and Cynthia Solomon launched the LOGO programming language and environment (see Figure 3 below). LOGO was one of the first high-level computer programming languages, and Papert, a protégé of Jean Piaget, designed the language and programming environment to embody social constructionist principles. According to Papert, children built knowledge through the construction of their programs. He continued to design and implement LOGO, creating what he deemed a social-constructionist sandbox. He believed that children are "active builders of their own intellectual structures" (1980, p. 19), where children could learn, apply, and come to know worldly concepts and things through the process of writing programs: the "child as epistemologist." Accordingly, he claimed that such material and symbolic activities fostered both what we are calling computational literacy and conceptual skills. According to Papert, the two are inseparable.

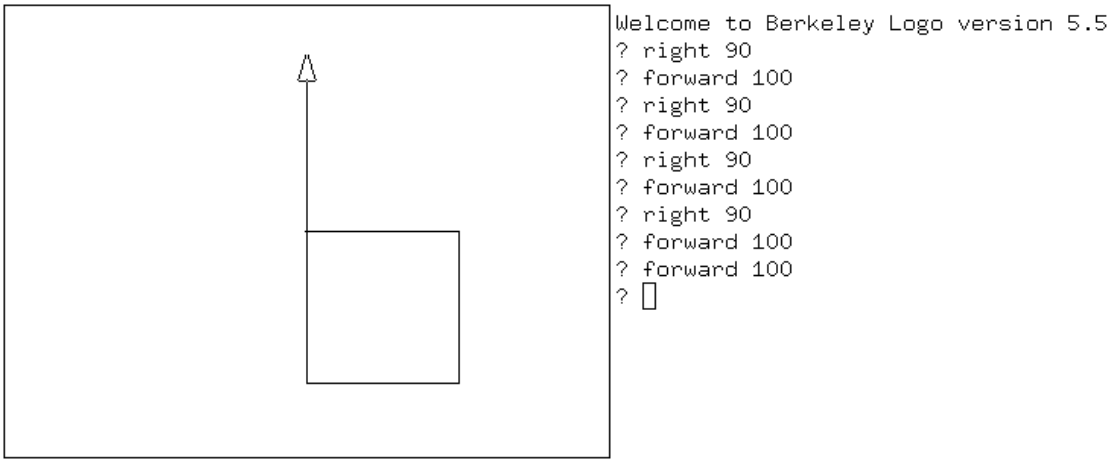

Figure 3. One of many iterations of the LOGO programming environment: source code on the right, result from source on the left (Wikipedia, LOGO).

Inspired by Papert's LOGO, computer scientist Alan Kay was integral in the development of Smalltalk, an object-oriented computer language for children. Kay's vision for such a language, he claims, was heavily influenced by Marshall McLuhan's ideas about the implications of new media. In an influential article about user interfaces and interaction, Kay (1989) writes “Though much of what McLuhan wrote was obscure and arguable, the sum total to me was a shock that reverberates even now. The computer is a medium! I had always thought of it as a tool, perhaps a vehicle—a much weaker conception. What McLuhan was saying is that if the personal computer is a truly new medium then the very use of it would actually change the thought patterns of an entire civilization” (p. 124). The more immediate implication of the computer as a medium is that it needs to be fully and effectively integrated into everyday discourse and education. “If the computer is only a vehicle,” Kay writes, “perhaps you can wait until high school to give ‘driver’s ed’ on it—but if it is a medium, then it must be extended all the way into to the world of the child” (p. 125). Kay and his development team at Viewpoint Research Institute later launched Squeak Etoys (see Figure 4), a higher-level programming language and environment built on top of Smalltalk. Etoys served as Kay's next step into the development of a programming environment, where kids could create objects and write scripts for their

objects to model the conceptual work they were learning in schools—a new method of multimodal writing that incorporates writing scripts.

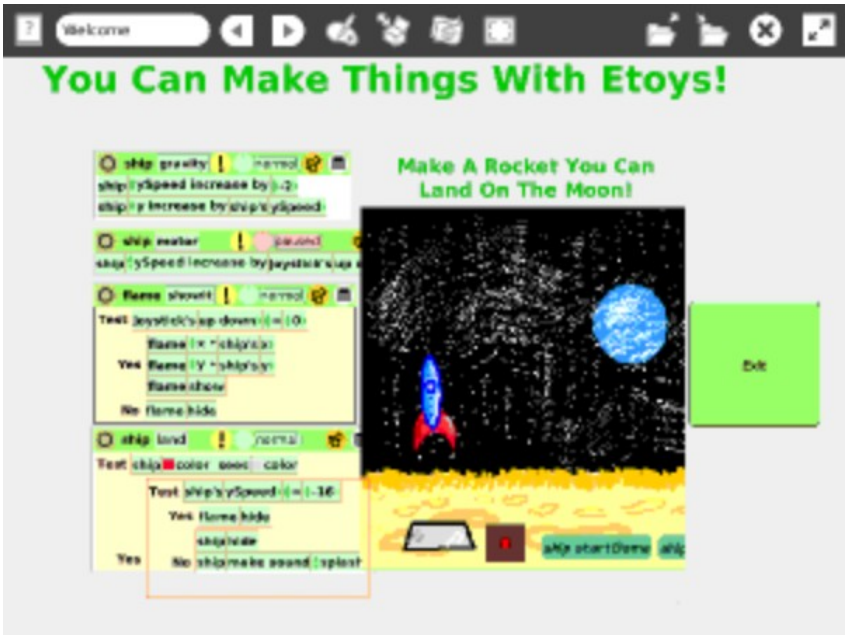

Figure 4. Drawing objects and writing scripts in Etoys (squeakland.org). (Animated GIF file on the website.)

Papert developed LOGO and Kay developed Etoys – two computational technologies designed for kids – but neither resorted to more drastic literacy crisis rhetoric to fuel their respective movements. Papert, in particular, never envisioned LOGO as the necessary means to accomplish all educational goals, nor was it a means for kids to forge careers. Instead, the philosophies driving Papert's LOGO educational programs (1987) were founded upon the rhetorical propensity to develop a new critical discourse for computing technology and learning by supplementing pre-existing pedagogical strategies. LOGO was "'just one more material'" (p. 25) to accomplish this goal, providing a representational system to build "'objects to think with'" (p. 24). He called this approach to learning, social constructionism, extending Piaget's theories of learning to include the material.

Papert argued that if students and teachers alike were even moderately fluent in LOGO, and they had access to computers and other supplies to carry out their projects, classroom spaces could become "messing places" (p. 24). In these experimental places, LOGO would simply become another material extension for student epistemic practices, rather than the sole means by which to construct knowledge. It is these educational philosophies that influenced the design of the Sugar OS, which was designed for One Laptop Per Child's (OLPC) XO laptop. It is also the OS we subscribed to in our outreach program.

Sugar is the OS designed by Walter Bender for the XO laptop. Bender, a MIT colleague of OLPC's founder Nicholas Negroponte, established Sugar Labs to opensource Sugar into contexts beyond the constraints of the XO. Sugar is a Linux distribution (Fedora), and we opted to use the USB bootable and writable "Sugar on a stick" (SoaS). We choose this version of Sugar, since it will run on any x86-based computer that can boot from a USB stick. In choosing SoaS as our technology, we hoped that we could more easily distribute these computing environments to children and their support systems

beyond the school and afterschool program to be used in the children's everyday contexts.

Sugar is designed with a different metaphor than most OSs. Instead of a "desktop" metaphor with windows and folders, Sugar is designed with a "views" metaphor, where learners conduct one activity on the screen at a time. When we learned more about Sugar, we were interested in testing its claims about its ability to facilitate collaboration and sharing within and between activities and resources. All files from the activities are saved automatically to a "Journal," which can then later be integrated into a portfolio activity to showcase projects. (See Figure 5 below to review the handout our graduate student team created for students to explore and learn the new Sugar OS environment.)

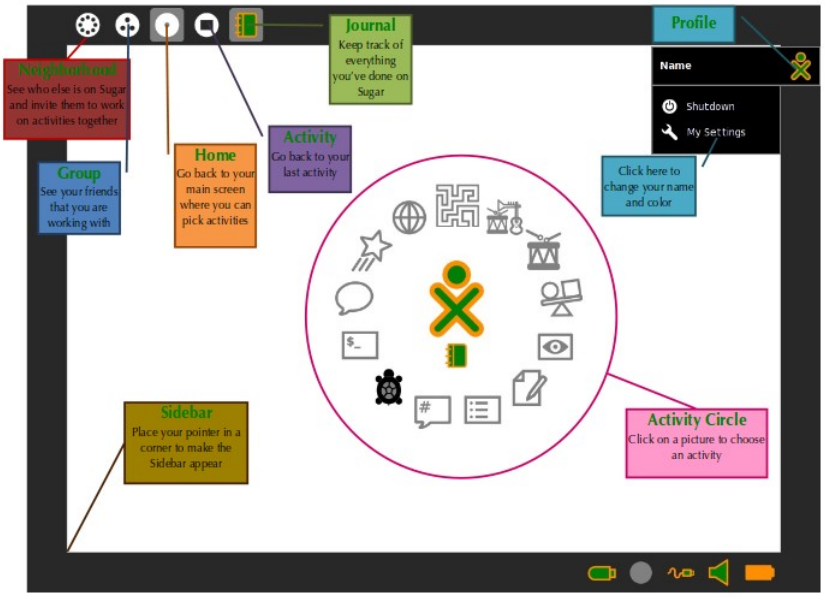

Figure 5. The Sugar Labs handout of the Home View in Sugar.

In the top-left corner of the screen, there are icons for the different views: Neighborhood, Group, Home, Activity, and the Journal. (Our chapter is designed with this UI design in mind.) The Neighborhood view enables learners to see who is connected to their (intra)network and what activities learners are working on and with whom. The Group view is similar, but enables learners to make groups and join activities in an ad hoc or planned fashion. The Home view (see above Figure 5) contains the activities, and the Activities icon simply will show learners what activities are currently running. The Journal view provides an organizable list of all of the saved activities.

We were particularly drawn to this different metaphor and collaborative opportunities for learners to work within the same activity. We were also intrigued by their design mantra, "Low floor, no ceiling" (Sugar Labs 2012), which emphasizes the range of skills learners either bring or (hopefully) develop within Sugar over time. This design principle carries into the activities, or applications, originally designed for Sugar, ranging from puzzle and maze games to built-in Python programming environments and View Source views for all activities and UI components. Bender, himself, ported a LOGO-like programming block environment into Sugar, which he called "Turtle Art," and the original Sugar OS includes Etoys as well. Sugar has a deep cultural and historical connection with the computational movements began by Papert and his lineage in the 1960s. In our next section, Crisis

Intervention Planning, we discuss the background and context of our program: Sugar Labs @ NDSU.

Crisis intervention planning

> *[C]hildren encounter LOGO in a particular way, in a particular relationship to other people, teachers, peer mentors, and friends. They don't encounter a thing, they encounter culture.* Seymour Papert (1987, p. 27).

Our after school initiative was not initially designed as an intervention in the coding crisis discourse; it was conceived of as an educational experiment with the potential for making an educational and social impact locally and globally. We wanted to connect the students from refugee families in Fargo with a global educational movement that might make an impact in their families' home countries, and we wanted to increase our own computational literacies so that we might be able to support a computer deployment in South Sudan at a school for girls established and run by a Fargo non-profit. From the very early planning stages, we also envisioned a city-wide effort that might take a small bite out of the significant amount of time kids in our community spend being consumers of digital products rather than makers of digital art, stories, and programs.

As we developed the project, however, we saw connections with, and without a doubt became interpellated by, the coding crisis discourse. We did not raise a lot of questions about the legitimacy of this crisis discourse because, as Trimbur notes, "[Crises] constitute necessary and enabling fictions that inscribe motives in educational policy and practice" (p. 281). We found ourselves drawing on the crisis discourse of "increasing Fargo's global competitiveness" and "increasing individuals' credentials" in order to sell our program to potential funders. We were aware of the potential to create a program that would replicate existing social stratification in the computing fields, so we emphasized in our rhetoric and practice the importance of building a diverse Tech Team with girls and under-represented minorities being over-represented in the program's composition.

A globally competitive or cooperative curriculum?

The crisis discourse of global competitiveness informed our curriculum as much the rhetoric of global cooperation. Philosophically, we wanted to offer a legitimate and rigorous introduction to computational thinking, which we framed for the school, ourselves, and students as central to 21st century literacy and success. But we also wanted to foster collaboration among the students and remind them of this projects connection to the larger OLPC global education effort. And finally, because we were approaching computational literacy from our humanities, computers and writing perspective, we aimed to foster expressive computing that would integrate design thinking and digital storytelling with an introduction to programming environments.

Practically and chronologically, we developed a curriculum that started with basic user interface instruction and activities, and then we used Etoys, a contemporary iteration of Alan Kay's SmallTalk, to teach digital storytelling in a programming environment. We let the students self-select into teams in order to encourage cooperative learning. We moved from narrative to math and design thinking by introducing students to Turtle Art, a contemporary iteration of Seymour Papert's LOGO. The programming environment in Turtle Art has a more gentle learning curve than Etoys, and the design challenges were more manageable in Turtle Art than Etoys, so we asked each student to complete their own challenges, but in a highly collaborative, sharing classroom setting. In the second half of our curriculum, we introduced a problem-solving environment called Physics and gave a lot of attention to the process of breaking a complex task (designing a Rube Goldberg-like machine) down into stages. As with Turtle Art, students worked individually but within a collaborative environment. We returned to

Etoys and a race-car game development project in an effort to develop their programming knowledge further, but this unit pushed us and the students to the edge of our computational literacies. We asked students to do a complex level of coding that they were not quite ready for, but the results were some interesting forms of resistance. We created handouts for the kids, which we called "homeplay," to encourage home use of SoaS, particularly between the two seven week sessions. Our curriculum can be found in one place on our project website, or broken out by Activity below:

- SoaS booting reference materials to take home: How to boot and What is the BIOS?;
- Physics - Level 1, 2, 3, and 4;
- Turtle Art - Week 1, 2 (see Figure 6 for an example challenge);
- Etoys - Week 1, Etoys at Home handout, and Waveplace Etoys tutorials (see Figure 7 for an example demo);
- Tech Team Homeplay given to students prior to their winter holiday break.

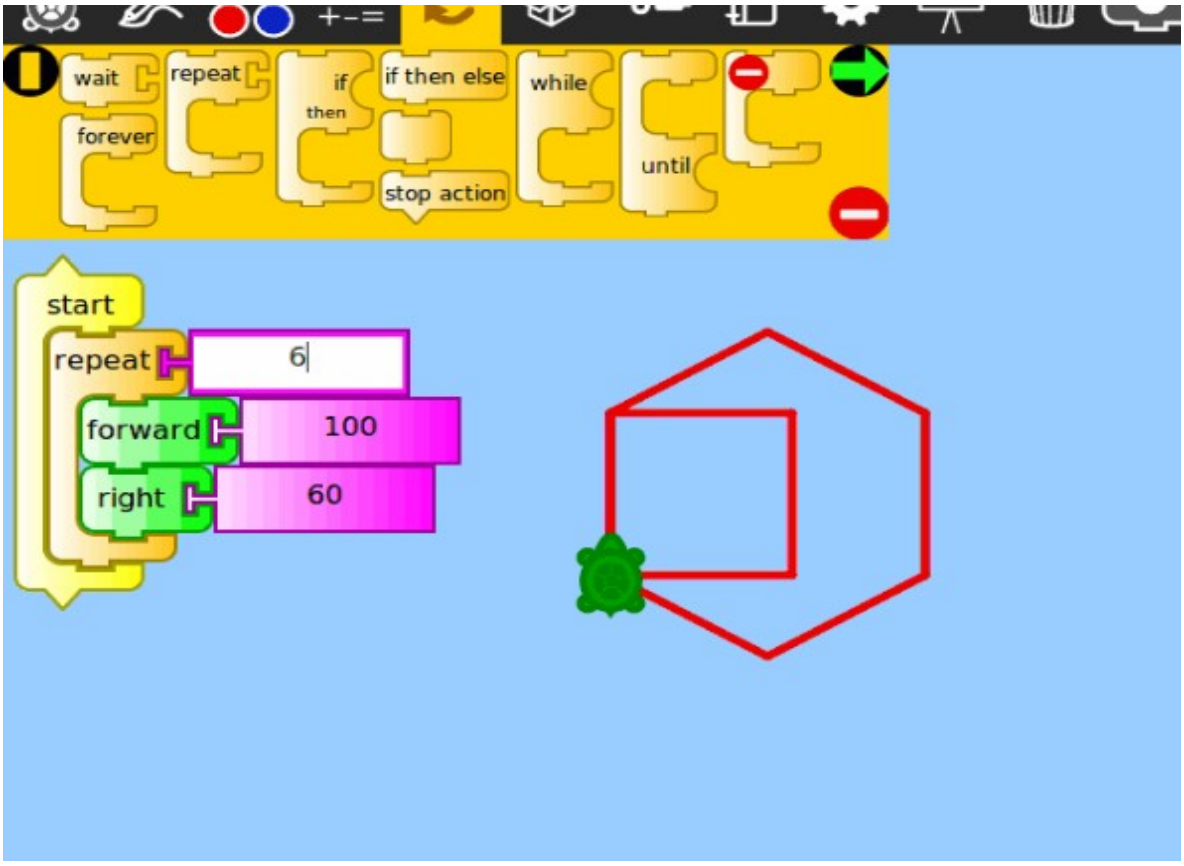

Figure 6. An example set of shapes that we challenged our students to create using the Repeat block during the first session with the Turtle Art activity. (Animated GIF file on the website.)

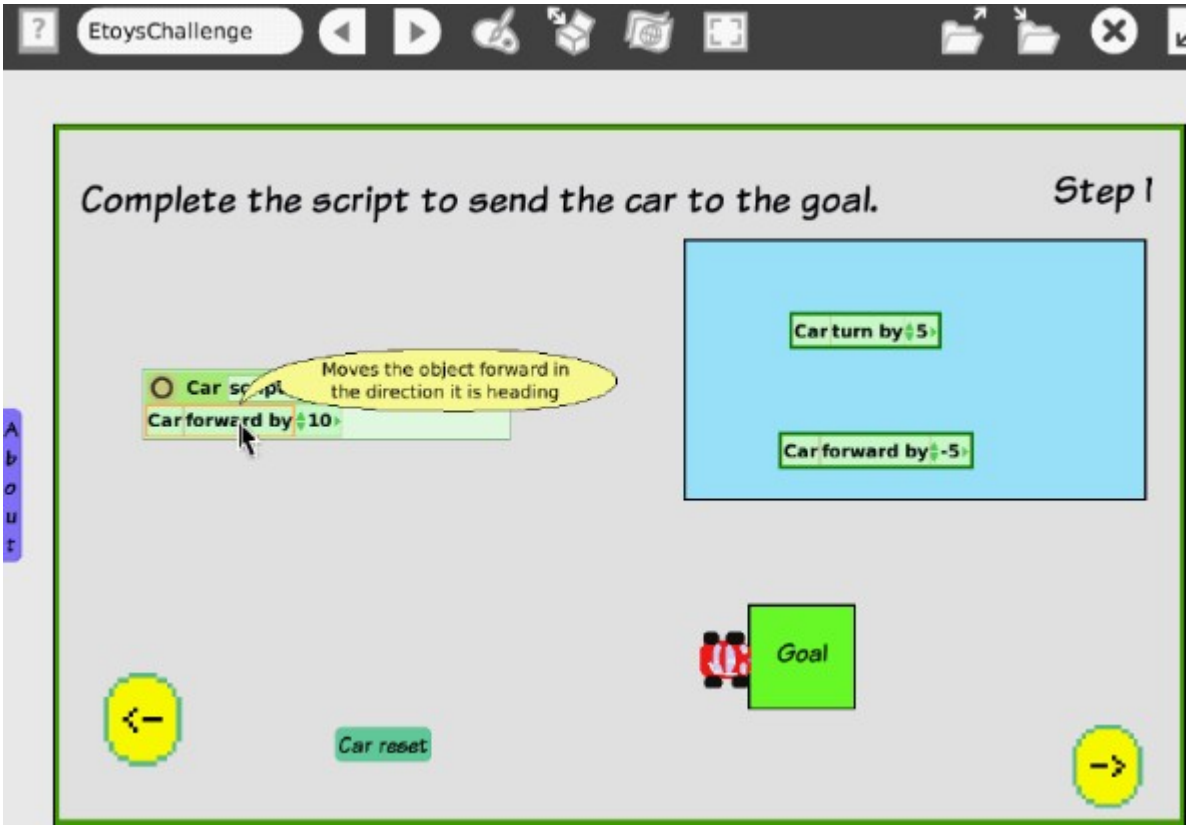

Figure 7. An example demo that helped students learn how to compose scripts during our first session of Etoys. (Animated GIF file on the website.)

With all of our units, we tried to provide sufficient scaffolding to enable success and completion of the activity, but we also tried to leave units open-ended enough for students to problem-solve, collaborate, and debug. Our goal was to help students develop the individual skills identified as computational thinking skills, but to develop those skills in a collaborative, social environment that we think matches the contemporary workplace—including the progressive nonprofit organization.

Individual credentialing or collective action . . . on a stick?

Because our initial goal was to connect local students to a global education effort that might resonate with them, and prepare ourselves for a possible computer deployment to South Sudan, we were trying hard to resist the privatization of literacy Trimbur identifies as central to a crisis discourse. Our public goal, however, also led us to use “Sugar on a Stick” rather than Scratch—another MIT platform for learning to code—or Kodu, the Microsoft programming for kids platform that might have lead to collaboration with our local Microsoft campus. “Sugar” is the name of the original operating system designed for the XO computer, the green and white laptop at the heart of the One Laptop Per Child Foundation. Sugar software designer Walter Bender parted ways with OLPC in 2008 (Lohr 2008), but continues to work on Sugar via Sugar Labs as an open source Linux operating system available for download and use on a “stick,” the colloquial name for a USB drive (also known as a thumb drive) as small as 1 GB, although we used 4GB sticks. Using Sugar gave us access to a number of programs (Etoys, Turtle Art, Physics, Implode, Maze, Typing Turtle), not just one programming environment, so we were able to design a curriculum with diverse activities.

We thought the SoaS would be an important part of our crisis intervention plan because it would be an affordable and ubiquitous technology we could give to all 200 students in the school. Like Code.org, our grand vision was to give every student in the school, and eventually the community, a way to develop computational literacy, but as we will explain in our analysis section, SoaS did not live up to its potential. Even we, two early technology adopters, found out that we did not have the time and expertise to customize and make enough sticks to sustain our program, so we hired an undergraduate student in computer science to provide that service. He typically had a 25% failure rate when making the sticks, and we ran into stick failures on a regular basis in the classroom, so we required fresh USB drives to be made throughout the year.

Trying to avoid social stratification

Our collaborators at the elementary school were open to our crisis intervention because they, like us, had invested in the crisis discourse about 21st century learning. The school had close to a 1-to-1 student / laptop ratio, so they were open to using their technology extensively and creatively. The technology specialist recruited a diverse "Tech Team" balanced by gender (7 boys and 6 girls) and race (6 white and 7 non-white) and only slightly skewed in favor of 5th graders (eight 5th graders, four 4th graders, one 3rd grader) to try out the technology.

Table 1.

Tech Team participants during the 2011-2012 year at Sugar Labs @ NDSU.

Student (Pseudonym)	Grade	Gender	Race	Year in program
Victoria	5	Girl	White	1
Sophia	5	Girl	White	1
Alayna	5	Girl	White	1
Jonny	5	Boy	White	2
Will	5	Boy	White	1
Alex	5	Boy	Asian	2
Carter	5	Boy	More than one race	2
Jacob	5	Boy	Black African	1
Jackson	4	Boy	Afro-Caribbean	1
Austin	4	Boy	White	1
Alexa	4	Girl	Black African	1
Maya	3	Girl	Black African	1
Faith	4	Girl	White	1

The ultimate goal was to have these students be classroom experts if / when we developed classroom uses for the school. We wanted to make sure that the program did not involve just the usual participants—boys, white or Asian—but even though we achieved a balanced team, gender, race, and age-groupings significantly influenced the social dynamics, as we will explain in the analysis section.

We ran a one-day summer workshop prior to the 2011-12 school year for teachers interested in Sugar, and teachers were invited to see what the Tech Team was doing with Sugar on any of the 14 days we were in the school. We invited the parents/guardians of the Tech Team students at the end of the first block of sessions to provide a way for the students to showcase their work and for us to share information about the program with parents, and ultimately encourage more home use. We knew that in order to make any kind of meaningful crisis intervention, the students would need to continue their computational literacy development at home and ideally share their knowledge with family members and friends.

Crisis intervention analyzed: Some failures and successes

For our analysis, we are going to combine two of the 3 coding crisis themes we have been examining: global competitiveness and individual credentialing. Considering the scope of this chapter, responding to Trimbur's themes, we are utilizing *preliminary findings* from our video and fieldnote data. Yet, we provide a glimpse into the experiences at Sugar Labs @ NDSU, touch on students' responses to our goals, as well as their individual competitiveness and cooperation. We are currently conducting a more thorough analysis of the data for further publication. However, what emerged much more clearly over the 14 weeks were the ways in which early signs of social stratification emerged. The SoaS itself, while meant to be an instrument of democratization, turned out to be an instrument of stratification. The mixed, balanced Tech Team was meant as a social technology for encouraging persistence and socially diverse access, but the loss of one member significantly affected another member of the Tech Team. The 2 Black African girls who quit are exactly the kinds of smart, talented students we and many efforts are trying to encourage and support.

Global competitiveness and individual credentialing

Neither the rhetoric of global economic competitiveness, nor the rhetoric of global education effort, seemed to motivate our students. Their reaction is understandable and appropriate: these elementary-aged children wanted to have fun and mess around with the computers. We introduced the Sugar Lab's connections to OLPC, and we brought in XOs to try one day (much too small and slow for these American kids who were working on slightly larger and slightly faster netbooks). We regularly asked them to reflect on what they were learning, but Kevin and one of the students fell into a familiar exchange that was clearly captured on video (refer to Figure 8; a video on the website). Kevin asked students to reflect on what they had just accomplished and learned with Physics, and then he looked right at one student, Jacob, and said, "you're just going to write, 'I had fun, but I didn't learn anything' aren't you?" Jacob responded with a "yes" and Kevin laughed. Yet, considering the environment and context, our afterschool program with no standards to meet, the emphasis on expressiveness and having fun most likely took precedence over preparing 5th graders for a globally competitive job market.

Jacob's easy-going attitude was not shared by all the students, all the time. One high-performing student, Jonny, brought his stage two Physics creation up to Chris to show him that he had achieved a very tricky balance for the objects on his screen, but Chris spotted the "pins" the student had put on his objects to keep them in place. Jonny sheepishly turned away, although Chris was laughing and the student exhibited a sort of cartoonish anger. Another student, Carter, working on the same activity got up quickly from his chair, raised both arms while carrying his laptop in his right hand and shouted out

to the room, “I completed it!” Trying to show completion of a task and showing excitement about completing a task are by no means evidence that students considered themselves on the road to global competitiveness, but these boys did exhibit fairly typical gendered, boyish competitiveness and seemed to take pride in being able to complete tasks quickly and effectively. Yet, these small gestures of sharing accomplishments often generated more spontaneous collaboration and expressiveness between students.

One of the girls, Victoria, also exhibited pride in completion and a really high level of persistence with tasks. She was consistently one of the most creative and engaged tech-teamers, but she was also one of the more reserved with her victories. As illustrated in video from a Turtle Art activity (see Figure 6 below), Victoria is working through one of the shape challenges, while simultaneously learning about all of the different blocks and variables. She says quietly to herself, "I don't know what I did. I'm sorry, turtle, I don't know what I did." Yet, after a few instances of trial-and-error, she celebrates her accomplishment with her friend by clapping and saying, "Oh, yeah! I made a bigger one!" Rather than announcing her success to the room, Victoria's more reserved celebration and creative computing elicited and engaged her friend, Sophia, throughout the 14-week program. Both were trying to solve Turtle Art challenges, and Sophia often watched, consulted, and followed her stronger peer, Victoria, throughout the entire process. As seen in the video (Figure 9; a video on the website), the two often moved back and forth between their screens, consulted our curriculum materials, and, in this case, they may also have been eavesdropping on some verbal coaching Chris was providing to a third student, Alayna (offscreen), at the same table.

During our weeks working in Etoys (refer to the time after 2:25 in Figure 7), where students composed scripts for the objects they created, we observed moments of collaboration and creative computing. Yet, upon review of our data, we also observed moments where we (Kevin and Chris) unintentionally invoked credentialing values through our student interactions. We introduced Etoys by first asking students to complete the provided tutorials within the Etoys activity (refer back to our curricula in the Crisis Intervention Planning section for examples). During the second session in Etoys, we challenged students to draw an object and compose a script for it after watching a tutorial on Etoys's tile-based scripts and features. On our final day in Etoys, we learned about variables and showed them a tutorial to write a script for a slider object to control, i.e., change a desired variable (rate of speed, y-axis location, etc.) of another object, that they had previously painted from another week. From there, we challenged them to take this knowledge and apply it to a joystick object, which takes 2 variables instead of just one. While Chris provided the initial demonstration of the slider and example joystick, many students followed along, except for Alex and Carter.

At approximately 3:20 of Figure 8, Kevin and Chris respond to Alex and Carter's accomplishment for the day, which was their recreation of one of the demo car's that is able to recognize a "track" to travel along. After reviewing the video data for this day, it is clear that Kevin and Chris misunderstood what Alex and Carter were attempting to share with them. To Kevin and Chris, they perceived a deviation from the challenge for the day. Alex and Carter, as evident in the video, were frustrated and resistant to our attempts to redirect their mindset to conform to our curriculum. What Kevin and Chris missed in this moment was how Alex and Carter collaborated for over 45 minutes on this project, and when they recreated a functioning car and track simulation, they also attempted to add a "nitrous" speed-booster feature to their car. Alex and Carter were proud of their work and creativity, but Kevin and Chris found themselves locked into the challenge of the day and consequently overlooked the value Alex and Carter placed in their accomplishment.

One student stands out as a model for what we are trying to accomplish; although the fact that individual rather than collective success emerged might mean that our program tilted in the direction of

individual credentialing. On the first day of our program in 2011, Jackson, an initially reluctant attendee with limited aptitude for computers and a big appetite for after school snacks, asked if he could lick the Sugar on a Stick. By the end of 14 weeks, our cooperating teacher told us that Jackson had been transformed from a docile student into an academically engaged student. We were told that other kids would ask him about Tech Team and what he was learning; we saw Jackson successfully coach his peers on the use of Sugar when we introduced it to all the fourth grade students. He didn't receive any official credential from his participation on the Tech Team, but from what we were told, Jackson received quite a bit of social capital through is participation, and he made greater investments in his education.

Social stratification emerged throughout the program

We feel more confident in reporting that even though we designed the program to counter-act the social stratification that already exists in the computer and high tech industry, and we tried to use a technology that is touted as a universally accessible technology, the most obvious result of our program is that we saw social stratification happening because of the hardware and the social networks at home and in school.

In choosing SoaS as our hardware-software combination, we hoped we were choosing technologies that would disrupt the students' understanding of computing. We were able to get students to think about operating system metaphors (Sugar, Windows and Desktop), and understand more clearly the separation between hardware and software as technology. We also hoped that we were choosing a technology that could move easily between school and home. We found, however, through our persistent efforts to encourage home use of SoaS that both technical and social barriers emerged. Sugar on a Stick requires that a computer, upon start up, recognize that an operating system is present on the USB drive, and that the computer chose that operating system over the hard drive's operating system (most likely Windows). Only three of thirteen students were able to get the SoaS to load on their computers at home, and most seemed to have their computers booting directly to their hard drive, with the boot order not set to look for a USB drive that could function as the operating system. Two of the students ran into social barriers: a father in one case and an older brother in the other case. These men seemed to control access to the family's computer and would not allow the USB drive to be used. We aren't suggesting this refusal was unreasonable; their legitimate concern might have been computer viruses. Only one student reported regular use of SoaS at home; based on what we knew about the families, he had the most familial support, in the form of a grandfather with significant computer experience.

We also saw in the classroom the kinds of social dynamics that seem to hinder the field of computer science and would hinder the widespread development of computational literacy. Based on the most common attendance of students, the typical table groupings were:

- Four white girls working together.
- Three boys working together: one white, one Asian, one mixed race.
- Four black African students (two girls, two boys) working together.
- The boy with the highest aptitude and interest in computers usually worked on his on.

When one of the black African girls quit, Maya, her best friend in the class, Alexa, continued for three more weeks, she continued to do good work, but she stopped smiling and communicating with us and the other kids. Alexa started sitting by herself, as seen on the video during a day when most of the students were working at the same table. Alexa resisted our attempts to integrate her, and eventually stopped coming. We saw her again on "Sugar Day," the wrap-up to the program when all the fourth

grade students in the school were introduced to SoaS; she was smiling and helping out, even though she had left the Tech Team. Whatever pleasure or challenge she got from working with Sugar activities, that pleasure was not sufficient to keep her coming back. When given a chance to share her knowledge and skills with classmates on "Sugar Day," Maya was much more like the student we saw during the first seven weeks. Anderson et al. (2008), in their search of the literature about gender and Information and Communication Technology (ICT) education, identify "peer support," or lack thereof, as one of the five key factors that contributes to the high attrition rate of girls from ICT subjects and professions. Maya didn't need support in order to get the work done; she just needed social support, she needed a close friend to share her interests and activities.

If efforts to teach computational literacy do not pay closer attention to the social dynamics of their classes or program, those who have formed a strong relationship with the computer, not necessarily other students—likely, but not necessarily, boys—will more likely benefit from these educational efforts. The students most at risk of falling behind—if that is what is happening—will be the students who are interested in classes or after school programs primarily for the social dimensions—likely, but not necessarily the girls. Volman et al. (2005) examined ICT preferences and skills among Dutch boys and girls, as well as majority and ethnic minority groups, and found that girls in their study preferred communicative and creative applications while boys preferred programming and games. Ethnic minority students used computers less frequently than the majority students, and tended to practice what they were being taught in school, rather than emailing friends or surfing the web. These findings, along with our preliminary findings, suggest that the social dimension needs to be the strength of a strong computational literacy effort—a social dimension that can draw in a wide range of community members, and support them in their computational literacy development.

Our analysis is of course limited in scope and time frame, due to the focus of this edited collection.[2] The two black African girls who quit in 2011-12 showed up for a few sessions in 2012-13 but did not persist. Our success story from 2011-12, Jackson, was a regular attendee in 2012-13, but became victim of SoaS's fickleness. He tried to use the same USB drive he finished 2012 with, but it failed, and he had particularly bad luck throughout the year with failing drives. By the end of the 2013 school year, we began exploring Scratch because Scratch 2.0 was released as a web-based platform, making it potentially more usable than SoaS at both home and school. We also saw that our efforts had run their course because we were not able to secure additional funding, the SoaS personnel were moving to new projects, and the work needed to properly support computational literacy, we learned, will be monumental. We have not given up on the dream of building a smarter computing culture in our community, but for now, we have put those efforts on hold.

Conclusion: subscribing to computational literacy

Rhetoricians are ideally positioned to engage literacy crises discourses, including the coding crisis discourse we have taken up. As Trimbur has argued, literacy crisis discourse often masks values and heightens a sense of crisis where one might not exist. We know that one of two dominant narratives are likely to inform the crisis discourse, and we can pay attention to additional narratives, like the dehumanizing anxiety narrative, that emerges with new stories. We can, however, also capitalize on a crisis discourse in order to accomplish our own goals and enact our own values, which are distinct from the dominant narrative. But as we have found out and acknowledged, that is a difficult move to make.

2Findings regarding our main research questions surrounding the the SoaS technology, and it as a mediational means in relation to computational practices, will be published elsewhere.

Rhetoricians do not have to limit their engagement with literacy crisis discourse to a rhetorical engagement, and we can't afford to make only broad claims that try to sweep the crisis discourse away. Even in the case of the coding crisis discourse, computers and writing scholars have a potential role to play. We can develop curricula and programs for storytelling and game development that introduce both teachers and students to computational literacy. Some of Trimbur's recommendations for strengthening the educational opportunities for economically disadvantaged students—free tuition and student stipends—are off the table in the current political and economic milieu (p. 294). That means that we need to be engaging with the students in K-12 environments who are the most disenfranchised by the current educational system.

As preliminary findings from our own research indicate, we also need to recognize that digital divides still exist within American communities. In particular, **access** is more complex and uneven than haves vs. have-nots, since some of the families of our Tech Team students owned computers, yet acted as gatekeepers in their very own homes. Furthermore, we need to be attentive to the social dynamics at school and in afterschool programs, because the gendered and peer patterns that persist in computer science and professional programming domains were visible to us among 4th and 5th graders.

The "program or be programmed" sound-bite and Code Year and Code Hour initiatives have functioned as megaphones for an apparent coding crisis, but the single-year and single hour solutions are magical quick-fixes Trimbur identified as likely to emerge in a literacy crisis discourse. We need to recognize that the 50 years of effort to build computational literacy practices are in fact still providing us with the tools and the imagination to tackle this challenge in a variety of ways. And finally, Trimbur's work should encourage us to pay close attention to who is being educated to be programmers, for what reasons, and to what ends. Through Sugar Labs @ NDSU, we were trying to ensure broad access to and participation in the full range of expressive powers available to students, teachers, and the community today, but we remain mindful of the need to not simply train our students for low-paying cubicle work or accommodate them to the machines. Rather, we seek to engage in literacy education with the goals of broadening access and participatory opportunities. We also hope that our work helps others pinpoint and refute the autodidact, bootstrap appeals of the coding crisis. To accomplish this, we need to think and value technosocial initiatives beyond Code Year programs and towards developing sustainable, flexible, and multiple forms of computational literacies.

List of Tables

List of Figures

References

Anderson, N., Lankshear, C., Timms, C., & Courtney, L. (2008). "Because it's boring, irrelevant and I don't like computers": Why high school girls avoid professionally-oriented ICT subjects. *Computers & Education* 50.4: pp. 1304-1318.

CODE: Home. (2013a). Code.*org*. Retrieved from http://www.code.org.

Code.org. (2013b) What most schools don't teach. YouTube. Retrieved from https://www.youtube.com/watch?v=nKIu9yen5nc

Codecademy: Home. (2012). *Codecademy.com*. Retrieved from http://www.codecademy.com/.

Codecademy: Code year. (2012 Jan. 1). *Codeyear.com*. Retrieved from http://www.codeyear.com/.

Code year [Screen capture of codeyear.com]. (2012) *Codecademy* [Author] Retrieved Nov. 20, 2012 from http://www.codeyear.com/

Computer [Modified PNG icon]. (2008) Sugar Labs [Producer] Retrieved Nov. 20, 2012 from http://git.sugarlabs.org/projects/sugar-artwork

Computer-XO [PNG icon]. (2008) Sugar Labs [Producer] Retrieved Nov. 20, 2012 from http://git.sugarlabs.org/projects/sugar-artwork

doThings.gif [Animated GIF]. (1990) *Squeakland.org* [Producer] Retrieved Nov. 20, 2012 from http://squeakland.org/images/doThings.gif.

Google. (2012 Jul. 4). Exploring computational thinking. Retrieved from http://www.google.com/edu/computational-thinking/index.html

Google's computational thinking [Screen capture from original website]. (2010) Google [Author] Retrieved Nov. 20, 2012, from http://googleforstudents.blogspot.com/2010/10/exploring-computational-thinking.html.

Google's RISE Awards [Screen capture from original website]. (2012) Google [Author] Retrieved Nov. 20, 2012, from http://www.google.com/edu/rise/index.html.

Gold, M., ed. (2012). *Debates in the Digital Humanities*. Minneapolis, MN: University of Minnesota Press.

Kay, A. & A. Goldberg. (1977). Personal dynamic media. *IEEE Computer*, 10, pp. 31-41.

Kay, A. (1984). Computer software. *Scientific American*, 251, pp. 53-59.

——. (2001). User interface: A personal view. In R. Packer and K. Jordan (Eds.), *Multimedia: From Wagner to Virtual Reality*. New York: Norton, pp. 121-131.

Lohr, S. (29 May 2008). Why Walter Bender left One Laptop Per Child. *New York Times.* Retrieved from http://bits.blogs.nytimes.com/2008/05/27/ why-walter-bender-left-one-laptop-per-child

Maney, K. (29 May 2014). Computer programming is a dying art. *Newsweek*. Retrieved from http://www.newsweek.com/2014/06/06/computer-programming-dying-art-252618.html

McPherson, T. (2012). Why are the Digital Humanities so white? Or, thinking the histories of race and computation. In M. K. Gold (Ed.), *Debates in the Digital Humanities*. Minneapolis: U of Minnesota P, pp. 139-60.

n.a. (2012). Sugar Labs educational nonprofit celebrates digital learning day with two Google code-in grand prize winners. *Sugar Labs*. Retrieved from https://www.sugarlabs.org/index.php?template=press&article=20130205&language=english#20130205

National Science Foundation. (2012). Computing education for the 21st century. Retrieved from http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503582

Negroponte, N. (2006), Nicholas Negroponte on One Laptop Per Child. *TED Talk*. Retrieved from http://www.ted.com/talks/lang/en/nicholas_negroponte_on_one_laptop_per_child.html

——. (Dec. 2008). Nicholas Negroponte on One Laptop Per Child, two years on." *TED Talk*. Retrieved from http://www.ted.com/talks/nicholas_negroponte_on_one_laptop_per_child_two_years_on.html

Nicks, D. (June 19, 2014) *Time*. The ambitious plan to teach 100,000 poor kids to code. Retrieved from http://time.com/2901198/computer-code-van-jones-prince-yeswecode/

NPR Staff. (Jan. 25, 2014). *All tech considered.* Computers are the future, but does everyone need to code? Retrieved from http://www.npr.org/blogs/alltechconsidered/2014/01/25/266162832/computers-are-the-future-but-does-everyone-need-to-code

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

———. (1987). Computer criticism vs. technocentric thinking. *Educational Researcher*, *16*, pp. 22-30.

Rushkoff, D. (2010). *Program or be programmed: Ten commands for a digital age*. Berkeley: Soft Skull Press.

———. (2012, Jan. 12). Learn to code, get a job. *CNN.com*. Retrieved from http://www.cnn.com/2012/01/12/opinion/rushkoff-write-code/index.html

Salter, A. (2012, January 10). New Year's resolutions: Learning to program. Chronicle of Higher Education: Prof Hacker. Chronicle.com. Retrieved from http://chronicle.com/blogs/profhacker/code-year/37845

Sample, M. (2012, May 19). 5 BASIC statements on computational literacy. Sample Reality. Retrieved from http://www.samplereality.com/2012/05/19/5-basic-statements-on-computational-literacy/

Sample, M, & A. Vee. (2012). Introduction to "The Role of Computational Literacy in Computers and Writing." *Enculturation: A Journal of Rhetoric, Writing, and Culture*. Retrieved from http://enculturation.net/computational-literacy

SITES. (2012). Program or be programmed: Do we need computational literacy in C&W? *Vimeo.com*. Retrieved from https://vimeo.com/42571756

Stolley, K. (2012). Source literacy: A vision of craft. *Enculturation: A Journal of Rhetoric, Writing, and Culture*. Retrieved from http://enculturation.gmu.edu/node/5271

Sugar Labs @ NDSU. (2011). A smart computing culture in Fargo. *Sugar Labs @ NDSU*. Retrieved

from https://fargoxo.wordpress.com/grant-proposals/proposal-2-sugar-on-a-stick-establishing-a-smart-computing-culture-in-fargo/

Sugar Labs @ NDSU. (2012). Google RISE proposal. *Sugar Labs @ NDSU*. Retrieved from https://fargoxo.wordpress.com/grant-proposals/google-rise-proposal-2012/

Trimbur, J. (1991). Literacy and the discourse of crisis. In R. Bullock & J. Trimbur (Eds.), *The Politics of Writing Instruction: Postsecondary*. Portsmouth, NH: Heinemann, pp. 277-295.

[Untitled screen capture from CODE.org]. Retrieved Feb. 12, 2013, from http://www.code.org/quotes.

Vee, A. (2012). Coding values. Townhall II Speaker. *Enculturation: A Journal of Rhetoric, Writing, and Culture*. Retrieved from http://enculturation.gmu.edu/node/5268

———. (2014). Computer programming as a literacy. *Composition and Literacy Studies, 1*, pp. 42-64.

Volman, M, van Eck, E., Heemskerk, I., & Kuiper, Els. (2005). New technologies, new differences. Gender and ethnic differences in pupils' use of ICT in primary and secondary education. *Computers & Education, 45*, pp. 35-55.