A Curriculum for Teaching Computer Science Through Computational Textiles

Kanjun Qiu
MIT Media Lab
High-Low Tech Group
Cambridge, MA 02139
kanjun@mit.edu

Leah Buechley
MIT Media Lab
High-Low Tech Group
Cambridge, MA 02139
leah@media.mit.edu

Edward Baafi
Modkit
Cambridge, MA 02139
ed@modk.it

Wendy Dubow
National Center for Women and Information Technology (NCWIT)
University of Colorado
Boulder, CA 80309
wendy.dubow@colorado.edu

ABSTRACT
The field of computational textiles has shown promise as a domain for diversifying computer science culture by drawing a population with broad and non-traditional interests and backgrounds into creating technology. In this paper, we present a curriculum that teaches computer science and computer programming through a series of activities that involve building and programming computational textiles. We also describe two new technological tools, Modkit and the LilyPad ProtoSnap board, that support implementation of the curriculum. In 2011-12, we conducted three workshops to evaluate the impact of our curriculum and tools on students' technological self-efficacy. We conclude that our curriculum both draws a diverse population, and increases students' comfort with, enjoyment of, and interest in working with electronics and programming.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education, curriculum.*

General Terms
Design, Documentation

Keywords
Computer science education, curriculum development, e-textiles, computational textiles, electronic textiles, LilyPad Arduino, tutorial, technological self-efficacy

1. INTRODUCTION
As the prevalence of technology increases in the world, the need for young people to acquire technological literacy and, we would argue, some programming experience is ever more important. Youth are currently introduced to computer science through design activities, such as video games and robotics, as well as through more traditional textbook curricula [15]. However, these existing avenues are very culturally narrow, targeting a limited set of interests; thus, these avenues attract a fairly homogenous and narrow population [11,15]. The emerging field of computational textiles (also called electronic textiles, or e-textiles) has shown promise in improving diversity, both in providing new and different educational materials for learning computer science, and in attracting a new and different population [2,4,18]. E-textiles integrate electronics and computers into soft materials such as fabric, allowing builders to sew electrical connections using conductive thread.

Computational textiles have been effectively used to introduce technology and computer science to students in ways that broaden the appeal of STEM (Science, Technology, Engineering, and Mathematics) fields [2,3,17,21]. Our investigation builds on this earlier work, focusing on enabling widespread and structured use of computational textiles in computer science and engineering education.

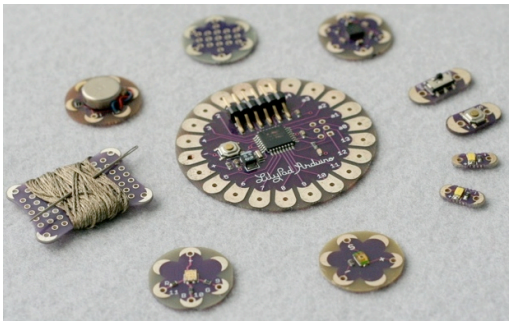

Figure 1. The LilyPad Arduino toolkit.

We use the LilyPad Arduino toolkit, shown in Figure 1. The LilyPad Arduino toolkit includes a microcontroller and a set of electronic components that makes creating electronic textiles objects accessible, while maintaining a focus on aesthetics. LilyPad components include speakers, lights, accelerometers, motors, and temperature and motion sensors, allowing for a diverse set of projects. In order to build an object, the creator uses conductive thread to sew a microcontroller (the LilyPad Arduino Main Board) and the relevant sensors and actuators to the fabric, and to each other. The conductive thread serves as an electrical connector, allowing the creation of the circuit while also physically securing the components. The builder then programs the microcontroller to receive sensor input and send instructions to the attached actuators.

2. RELATED WORK
Previous work, which we will now briefly survey, shows that the LilyPad Arduino toolkit and the construction of e-textiles make

electronics and programming more approachable and accessible, and improve students' perceptions of electronics and technology.

A series of workshops run by Buechley, Eisenberg, and Elumeze in high schools showed that working with LilyPad and e-textiles increased basic circuit and programming knowledge [3]. However, this 2007 endeavor to develop an e-textiles curriculum for high school students encountered logistical and technological problems: an early prototype of the LilyPad toolkit was difficult for novices to use, the classroom sessions were very short (55 minutes), and the curriculum was fairly unstructured. These features combined to prevent participants from completing satisfying projects [3]. Other early efforts, like that of Katterfeldt et al., focused on using e-textiles and the LilyPad Arduino toolkit to teach young people general concepts about technology and its relationship to other aspects of life and faced some of the same technical challenges [13][7].

More recent work has examined what students learn by building projects with the LilyPad Arduino toolkit, focusing particularly on the unique affordances of e-textiles. Kafai, Fields, and Searle for example, explored how e-textiles (and craft more generally) can increase the visibility and legibility electronics and programming concepts. They argue that learning can be enhanced by this integration of craft and technological processes [12]. The same group has also explored how technological learning can be enhanced and deepened when students are engaged in the aesthetic design of a project [8].

In short, a significant and growing body of research has shown that e-textiles can be used successfully in educational settings. However, the research has also exposed the challenges faced by researchers (and educators) in working with these materials.

3. MOTIVATION

E-textiles have remained largely inaccessible to educators and potential students [2,3]. The reasons for this include the fact that: 1) there is a dearth of educational materials (like well-designed activities and lesson plans); 2) no large-scale attempt has been made to use e-textiles to teach computer science and electronics (as a result, most educators simply aren't aware of e-textiles); and, 3) text-based programming tools are intimidating and challenging to use. As a result, e-textiles are prevalent in the hobbyist community [4], but have not yet made their way into classrooms, summer programs, or community centers on a large scale.

Because computational textiles are not yet accessible to educators, we developed a structured curriculum to teach computer science through a series of computational textile projects built on the LilyPad Arduino, targeting middle and high school students. The curriculum is comprised of a series of hands-on lessons in which students build projects, each of which emphasizes particular computer science concepts. Our goal is to pique students' interest in computer science by presenting computer science concepts through the lens of e-textiles.

To evaluate the effectiveness of the curriculum in building a more diverse computer science community, we taught a series of workshops, each based on a project in the curriculum. To determine whether student interest has been "piqued", we evaluated whether going through the curriculum project in each workshop increased students':

1. *Comfort with* building electronics and programming computers on their own
2. *Enjoyment of* building electronics and programming computers
3. *Interest in* learning more about electronics and programming

To ensure that the curriculum is practical and implementable, we also evaluated whether students were able to complete a project within the allotted time.

4. NEW TECHNOLOGIES: MODKIT & LILYPAD PROTOSNAP

Simple circuits—constructed from LEDs, switches, and conductive thread—can be built in 1-2 hour sessions that do not involve computers. In contrast, building computational projects—projects that involve programming—takes significantly more time. In early workshops, researchers found that constructing computational projects took at least a full week, five 6-7 hour days. The construction of a computational project requires that a student simultaneously learn sewing, design, electronics, and programming; it is difficult to complete a computational project without acquiring competency in all of these areas [2,3]. A significant contributor to these challenges, is the difficulty students face learning a text-based programming language. Here, we present two new technologies we have developed to improve the ease with which educators and students can engage with computational textiles.

4.1 LilyPad Arduino ProtoSnap Board

The LilyPad Arduino ProtoSnap board, designed in collaboration with Pete Lewis at SparkFun Electronics, enables students to experiment with programming before they build a physical project. The ProtoSnap board, shown in Figure 2, allows students to write and test out programs without simultaneously trying design and debug electrical circuits. In our previous experiences, we found that it was very difficult for students to learn how to program while they were learning how to design electronics and sew circuits together. The ProtoSnap board allows students (and educators) to decouple these activities.

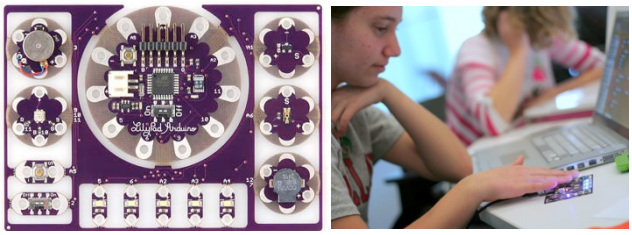

Figure 2. Left: the LilyPad ProtoSnap. Right: a student programming and experimenting with her board.

The board consists of a collection of inputs (two switches, a temperature sensor, and a light sensor) and outputs (five LEDs, a motor, and a speaker) that are pre-arranged and pre-connected to a LilyPad Arduino microcontroller board. It can be programmed as a stand-alone device. Students can use the board to learn how to write basic programs, learn how different input and output devices function, and prototype the behavior of their projects. The pieces can be snapped out of the board and stitched together when a student wants to build a physical project.

4.2 Modkit Programming Environment

A significant body of research has shown that many people find traditional text-based programming challenging [6]; in contrast, visual programming languages such as Scratch and Alice have been highly effective at introducing beginners to programming [6,14,19].

The Modkit programming environment, developed by Ed Baafi, was created to address this accessibility issue: it allows users to

program the LilyPad Arduino (and other Arduino boards) using a Scratch-like visual programming environment [1][20].

Modkit has three different interface views that correspond to different actions a user can take. The *hardware view* allows users to describe the configuration of their electrical circuit, to indicate which hardware components are attached to different pins on the LilyPad Arduino. The *blocks view* is a programming interface where users drag, stack, and nest code blocks to create a program. Finally, a *source code view* displays the C code that was generated by the blocks. Screenshots of Modkit's blocks view and hardware view are shown in Figure 3.

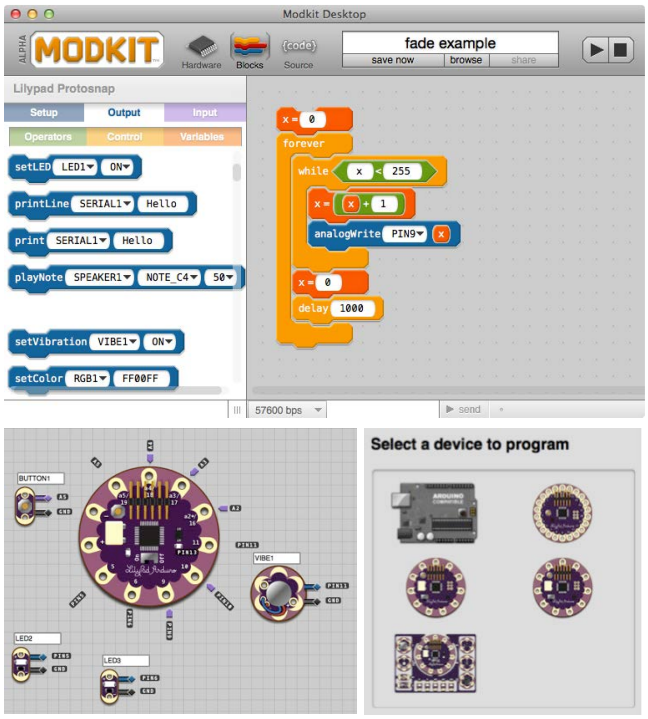

Figure 3. The Modkit programming environment. Top: the programming interface (software view). Bottom left: the hardware view. Bottom right: the opening screen, displaying different boards to choose from.

While other researchers have developed visual programming languages for the LilyPad Arduino [13], Modkit is unique in a few important respects. It was designed not only to be a visual programming language, but also to help people understand the relationship between electrical circuits and programs—a particular challenge for students building e-textiles.

When a user opens the Modkit environment she first uses the hardware view to design her circuit—telling Modkit which components are used in her design and how they are connected. For example, she may choose to attach an LED to pin 5 on the LilyPad and a temperature sensor to pin A2.

Then, when she opens the code view, it shows code blacks that can control each of the components she is using in her design. It would, for the configuration described above, show LED and temperature sensor code blocks that she could use in her program. If a user is employing a LilyPad Protosnap, she can select this board in the hardware view and the blocks view will present her with blocks to control each of the components on the board.

No other widely used programming environment captures these relationships between electrical layout and code. These relationships are often one of the most difficult things for novices to understand and keep track of in their projects. The creation and execution of code on a computer and the behavior of objects in the physical world exhibit no visible ties. Moreover, problems in e-textiles can be particularly hard to identify and correct since errors may be present in either the code or the electrical circuitry. Modkit helps students understand and manage these connections by making them a central and explicit part of the design and programming process.

5. CURRICULUM FRAMEWORK

Before describing the curriculum, we first define the framework that guided its development.

5.1 Educational Philosophy

We began with the foundational belief that a successful curriculum should engage students and provide a balance of structure and openness. A compelling activity should supply the guidance required to successfully build a project, while providing ample room for personal exploration, engagement, and expansion [10,16]. We also believe in providing a structure that helps learners experience early successes and supports multiple learning styles.

5.1.1 Learning by doing

Our approach is grounded in constructionist learning theory [22]. We presume that children are particularly engaged and effective learners when they are building personally meaningful artifacts. Thus, each lesson involves students in creative construction and encourages personal exploration and expression.

5.1.2 Encouraging early successes

To support early successes in each curriculum project, we designed the curriculum to be self-contained. That is, we assume no prior exposure to sewing, electronics, or programming; all the skills and information needed for completing projects is introduced sequentially in the curriculum. Moreover, our workshop experiences (which we will describe shortly) helped us identify appropriate levels of difficulty for each lesson. We have taken care to slowly introduce, as well as emphasize and revisit, particularly challenging ideas and processes in the lessons.

5.1.3 Supporting multiple learning styles

Turkle and Papert's articulation of epistemological pluralism [23] inspired our efforts to support different learning styles. They argued that learning materials should support different epistemologies—different ways of seeing and making sense of the world. To support epistemological pluralism, we provide multiple pathways for students to engage with important ideas. A circuit, for instance, can be represented in several ways:

1. Conceptual representation: As a conceptual diagram of a traditional electrical circuit.
2. Visual representation: As an illustration with LilyPad components and conductive thread.
3. Aesthetic representation: As an illustration in context of a project, showing how the circuit is stitched into the project and incorporated into its decoration.

By supporting personal creativity, designing for early successes and introducing concepts from multiple perspectives, we aim to build a curriculum that engages students' interest, and encourages motivated, independent learning.

5.2 Target Populations

Our curriculum was also designed with particular audiences in mind.

5.2.1 Target learning communities

First, we elected to target community organizations, after-school programs, and summer camps as pilot learning communities. These communities are generally receptive to experimental curricula, which makes the curriculum more easily adoptable. Moreover, activities in these communities tend to last for several hours at a time, as opposed to activities in classroom settings, which are usually broken into short (one-hour or shorter) time periods. When building computational textile projects, more progress in teaching and learning is often made when working in chunks spanning several hours [3].

5.2.2 Target educator profile

Second, we envisioned the ideal educator to be a teacher familiar with craft and/or art, rather than a teacher who has strong technical training. Because the curriculum is designed as an introduction to computer science, deep technical knowledge is not required to implement and teach the computational concepts. On the other hand, due to the strong emphasis on personalized and aesthetically driven computational projects, we believe that a teacher with strong craft experience is well positioned to support the design, hands-on construction, and iterative testing of student projects.

5.2.3 Target user audience

Third, we designed the tutorials with a middle and high-school audience in mind. Furthermore, we envisioned the educator as a facilitator of semi-independent learning as much as a direct instructor. Therefore, we committed to writing the curriculum in a voice that directly addresses the end-user, rather than telling the educator how to teach their students. This is intended to give the curriculum versatility. An interested student could use the curriculum as a book of tutorials, and work through them independently. Or, an educator who is unfamiliar with computer science could work through the projects as a student to learn the material, and then facilitate learning with her students.

5.2.4 Publishing as a self-contained book

Fourth, we committed to publishing the curriculum as a self-contained book, as well as to posting materials online. Presenting the book as a contained, physical object lowers barriers for entry by decreasing the number of steps needed to get started. This allows any educator or young person to pick up the book, order materials, and start building projects.

6. CURRICULUM CONTENT

We elected to present the curriculum as a series tutorials for building a set of increasingly complex computational textiles where each subsequent project builds on knowledge gained from earlier projects. This allows us to begin with a very simple project, encouraging early successes, and slowly progress into more difficult projects that involve more complex electronics and computer science concepts.

We approached the creation of the curriculum's content as an iterative design process: develop a project, test the efficacy of the project through an evaluative workshop, revise elements that were ineffective, retain effective elements, and add useful elements that were discovered in the workshop. This process was guided by our goal to increase students' comfort with, enjoyment of, and interest in electronics and computer science.

6.1 Computing Concepts

To establish important programming concepts for beginners, we examined introductory computer science textbooks including Python Programming: An Introduction to Computer Science, by John Zelle [24]; and Practical Programming: An Introduction to Computer Science using Python, by Campbell, Wilson, Gries, and Montojo [5]. We also looked at the high school AP Computer Science A curriculum, as well as college-level introductory computer science courses at MIT (6.00), Harvard, (CS50), and Stanford (CS 106A). The concepts we distilled from studying these curricula can be separated into three categories: High-level principles, more specific implementation-level concepts, and good programming practices. We believe that the structure and specificity of the learning goals we want to achieve improve the curriculum's effectiveness and impact.

6.1.1 Computing principles[1]

Procedural Thinking: Students should understand that a program executes sequentially, line-by-line. Students should understand control flow, how logical statements are used to structure and change the way a program executes.

Abstraction and Modularity: Students should understand how abstractions like variables and procedures can be employed to solve *classes* of problems rather than specific *instances* of problems. They should understand how this kind of abstraction enables programmers to focus on high-level structures and ideas, which helps them build complex software systems.

Computer Architecture: Students should be familiar with the basic components of computer architecture: memory, processor, inputs, and outputs. Students should understand that different parts of a computational system need to communicate with one another.

6.1.2 Computing concepts

Programming Process: Students should understand how programs are written, compiled, loaded into memory, and then executed.

Variables: Students should understand that variables are a way for the computer to store values in memory, and that variables are necessary when values need to be stored for later use. Students should also understand how variables make code more readable, maintainable, and flexible.

Control Flow: Students should understand how to control program flow using conditional and loop statements. Students should understand if-else and while statements along with comparisons between variables or values. Included in this understanding is a knowledge of Boolean values and how to use them.

Procedures: Students should understand how to use procedures to generalize and reuse code, how to employ procedure inputs to improve modularity and generalizability, and how to return values from procedures.

Data Structures: Students should understand that data structures provide ways to store data in a particular format in a computer's

[1] The knowledgeable reader will have noticed that many of the big ideas in computer science are not on this list. Among these are: computability, complexity, and recursion. The intention of our curriculum is to engage students in creative programming more than to enable them understand computers in an abstract way. Our hope is that students will go on to explore these and other big ideas in computing after encountering our introductory lessons.

memory. Students should understand how to create, populate, and use arrays (as example data structures).

User Interfaces: Students should understand how computers receive input from input devices and display information via output devices. They should understand that interface components are designed and design-able artifacts.

6.1.3 Computing practices

Incremental Development: Students should learn that code is best developed incrementally and iteratively, with frequent bouts of testing.

Debugging: Students should understand that debugging is a natural part of electronics and computer science, and should learn to expect to debug circuitry and code and to do so with a minimum amount of fear and frustration.

6.2 Curriculum Projects

After developing these principles, concepts, and practices we brainstormed computational textile projects that required mastery of different sets of them. We had three primary criteria for choosing projects. First, the project must implement at least one principle, concept, or practice. Second, the project must allow for extensibility, both in terms of physical construction and program functionality. Finally, the project should be fairly gender-neutral, to appeal to children of all genders. We now describe the projects and the skills and ideas they introduce.

6.2.1 Light-up Bookmark

The light-up bookmark is a simple circuit that has no LilyPad board (and therefore no microcontroller) and requires no programming. The bookmark has a single battery holder with one coin cell battery and one LED, connected by conductive thread. When the battery is inserted, the LED lights up. A sample bookmark is shown in Figure 4.

Figure 4. A sample light-up bookmark project.

Students should understand:

1. How to sew a simple running stitch and tie sewing knots.
2. How electricity flows through a circuit. Voltage and current.
3. How to sew circuit components to each other to form a functional circuit.
4. Debugging: How to identify and fix short circuits, loose connections, incorrect polarities, and other craft and electrical problems.

6.2.2 Interactive Monster

The interactive monster is a stuffed plush monster with a LilyPad Arduino board, a speaker, and an LED. The LilyPad is programmed to control the behavior of the speaker and LED.

An extension to the basic monster project adds two conductive paws, comprised of two pieces of aluminum foil that function as a resistive touch sensor. The extension describes how to program the monster so that its behavior changes when its paws are being touched. A sample monster is shown in Figure 5.

Students should understand:

1. Procedural thinking: How code executes line by line, in order. How the behavior of electronic components can be changed when code is changed.
2. Programming process: How code is written, compiled, loaded onto a LilyPad, and executed.
3. Outputs: How to control outputs programmatically.
4. Abstraction: How to use variables to avoid duplicating code, and to make code more readable and flexible.
5. Abstraction: How to create and use procedures. How these procedures make code more modular, readable, and maintainable.
6. Inputs: How to read sensor data and use it in a program.
7. Communication: How serial communication allows the computer to "talk" to the LilyPad. How to send data from the LilyPad to the computer.
8. Control flow: How to create condition-dependent behavior and change a program's execution flow using if-else statements.

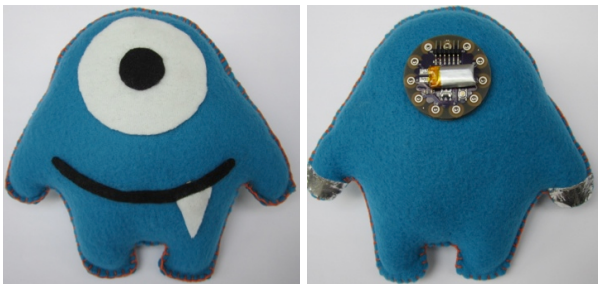

Figure 5. A sample interactive monster project.

6.2.3 Fabric Piano

The fabric piano has touch-sensitive conductive keys that are each connected with conductive thread to a pin on the LilyPad Arduino board. Capacitive sensing is used to make the keys touch-sensitive. The piano can play notes both on a LilyPad speaker connected to the LilyPad Arduino, as well as through a piano emulation program (written using Processing [22]) that runs on the computer. A sample piano is shown in Figure 6.

Figure 6. A sample touch-sensitive fabric piano project.

Students should understand:

1. Control Flow: How to use if-else statements and loops.
2. User Interface Design: How serial communication allows the LilyPad to control the computer, to function as an interface.
3. Abstraction: How input parameters make procedures more flexible. How to create and use input parameters..
4. Data Structures: How to create and use arrays. How to store and access values in arrays.

7. CURRICULUM EVALUATION

In order to evaluate the potential effectiveness of the curriculum suggested above, we ran three evaluative workshops in the fall of 2011 and the spring of 2012 with middle and high school-aged

students. We used these workshops to improve the relevance, appeal, and ease of implementation of our curriculum projects.

7.1 First Interactive Monster Workshop

Our first workshop took place in the fall of 2011 over the course of a weekend, lasting for 6 hours on both Saturday and Sunday. Our primary goals were to: 1) determine the feasibility and appeal of the activity; 2) understand the affordances of teaching computer science with Modkit and the LilyPad Arduino ProtoSnap; and 3) determine if participation in the workshop could positively impact students' technological self efficacy.

We recruited participants through our research group's mailing list, and through educators we had previously collaborated with. This session drew 16 participants, ages 13-16. 75% of these participants (12 students) were female and 6% (1 student) were underrepresented minorities.

Students completed two activities during the workshop: 1) an e-sewing activity with a LilyPad battery holder and LED (light-emitting diode), similar to the bookmark project in our curriculum; and 2) the interactive monster project from our curriculum.

Students began with the simple e-sewing activity. Then they were introduced to programming via the LilyPad ProtoSnap boards and Modkit through a series of exercises. Next, students programmed their desired monster behavior on the ProtoSnap boards, and then unsnapped the pieces to attach them to their monsters, securing electrical connections between components by sewing with conductive thread. Images from the workshop are shown in Figure 7.

Figure 7. Left: Two students collaborating while programming. Right: Completed monster project.

Students picked up programming concepts (such as conditional statements and looping) intuitively in Modkit. Students encountered significantly fewer frustrations while programming with Modkit than while programming with text-based tools. Students did not have to worry about using the correct syntax or spelling, and could instead focus on understanding higher-level programming concepts. The ProtoSnap board allowed students to see the results of their programs without first constructing a project.

All students completed both projects by the end of the session, though several students had not finished stuffing and sewing their monsters at the conclusion of the session. In post-workshop surveys, 85% of participants said they had created projects they were happy with. We also saw significant increases in students' self-reporting of technological self-confidence, engagement, and interest from the beginning to the end of the workshop session. A summary of these results is shown in Table 1.

7.2 Second Interactive Monster Workshop

For our second interactive monsters workshop, also held in the fall of 2011, we wanted to engage with a more diverse group of students to determine effectiveness of the curriculum across different educational and cultural backgrounds. We partnered with a Boston youth organization called Learn2Teach Teach2Learn to recruit students from around the city; as a result, this workshop was attended by 9 students ages 12-17, with 67% female (6 students), 56% Hispanic or Latino (5 students), and 38% Black or African American (3 students).

Students worked on the same two activities as in the first interactive monsters workshop; as in the first workshop, all students completed both interactive projects, but several did not finish stuffing and sewing their monsters. However, we modified the teaching of the curriculum to evaluate whether students could effectively transition between using Modkit's drag-and-drop block-based programming to understanding and generating C code using the regular Arduino programming environment. On the first day, students used LilyPad ProtoSnap boards and created Modkit programs that achieved the behaviors they wanted in their monsters. On the second day, we had students reprogram their monsters with similar behaviors by writing syntactically correct code in the regular Arduino development environment; students started with the example "Blink" program that is included with an Arduino installation, and modified it to produce their monster's behavior. Images from this workshop are shown in Figure 8.

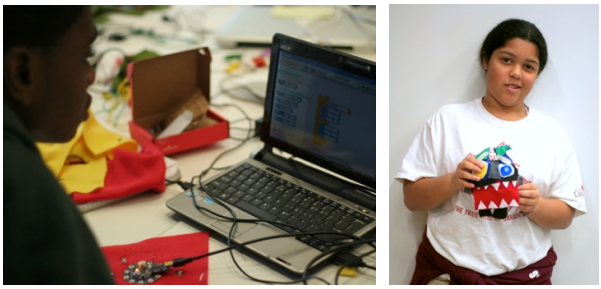

Figure 8. Left: Student programming his monster using Modkit. Right: Completed monster project.

When examining pre-workshop and post-workshop surveys for this session, we saw similar patterns of increased engagement and technological self-efficacy. Results for both workshops are shown in Table 1.

Table 1. Student attitudes toward programming and electronics in both interactive monster workshops.

	Workshop 1		Workshop 2	
Statement	**Agree before**	**Agree after**	**Agree before**	**Agree after**
I feel comfortable programming computers on my own.	25%	65%	56%	78%
I feel comfortable building electronics on my own.	25%	71%	67%	78%
I enjoy programming computers.	67%	88%	90%	100%
I enjoy building electronics.	73%	100%	78%	89%

7.3 Fabric Piano Workshop

Our third and final workshop with students took place in the spring of 2012. Like the previous two workshops, it was held in two days over the course of a weekend, for 6 hours each day. We recruited students through our research group's mailing list, through a university-wide mailing list targeted at educators, and through the Learn2Teach, Teach2Learn program. 11 students participated, ages 12-17, with 45% female (5 students), 9% Hispanic or Latino (1 student), and 55% Black or African American (6 students).

Students completed two activities during the workshop: 1) the same e-sewing activity as in the previous two workshops; and 2) the touch-sensitive fabric piano outlined in our computational textiles curriculum. Images of projects built in the workshop are shown in Figure 9.

Figure 9. Left: Completed fabric piano project. Right: Student testing fabric piano with LilyPad speaker.

In this workshop, we went one step further in examining the transition between Modkit and Arduino's development environment. On the first day, we taught students how to program blinking LEDs and how to play notes on the LilyPad speaker using Modkit and the LilyPad ProtoSnap board. This initial program was not directly related to the final touch-sensitive piano, partly because the ProtoSnap board does not easily afford capacitive sensing behavior. On the second day, we began again with the example Arduino "Blink" program, and walked students through modifying the sample code to incorporate an Arduino capacitive sensing library (CapSense), and then using the library's API to detect touch and control the piano's behavior. Students code the program's behavior, with guidance, after being introduced to basic programming concepts and the relationship between software and hardware through working with Modkit and the ProtoSnap board.

All students were able to create functional versions of both projects by the end of the workshop. In post-workshop surveys, 92% of participants said they had created projects they were happy with. Before the workshop, 27% of students agreed or strongly agreed with the statement "I feel comfortable programming computers on my own". 82% agreed after completing the workshop. Similarly, 18% of students agreed or strongly agreed with the statement "I feel comfortable building electronics" before the workshop and 73% agreed at the conclusion. Survey results for the workshop are shown in Table 2.

Table 2. Student attitudes toward programming and electronics in the fabric piano workshop.

Statement	Agree before	Agree after
I feel comfortable programming computers on my own.	27%	82%
I feel comfortable building electronics on my own.	18%	73%
I enjoy programming computers.	73%	91%
I enjoy building electronics.	100%	100%

7.4 Evaluation Conclusions

The survey results from our workshops demonstrate the potential of a computer science curriculum taught through computational textiles. The data shows that building the projects in our structured curriculum impacts builders' technological self-efficacy, leading to in an increase in students' comfort with, enjoyment of, and interest in programming and electronics. Moreover, students were able to successfully complete functional projects, and most reported having a positive overall experience. Finally, the students who self-selected to participate in our workshops were 63% female (22 students), comprising a significantly higher percentage than in traditional electronics communities, which hover at around 10% female [4,9]. This reveals the promise of computational textiles to diversify both the culture and population of existing computer science communities.

The workshops validate the effectiveness of combining Modkit with the LilyPad ProtoSnap board to introduce programming quickly and with minimal frustration. They alsoconfirm that students can successfully transition from building programs in Modkit to writing C code. Additionally, we explored the three options available to educators when using Modkit: in the first interactive monsters workshop, we taught students to program their monsters using only Modkit; in the second interactive monsters workshop, we had students fully program their monsters using Modkit, and then replicate and add functionality by programming in C using the Arduino development environment; finally, in the fabric piano workshop, we built one program using Modkit, and then programmed the touch-sensitive piano entirely in the Arduino development environment. In each instance, students had a positive experience, and students' engagement with programming and electronics generally increased.

Another noteworthy outcome from these sessions was how much we were able to accomplish in a weekend workshop with the LilyPad ProtoSnap and Modkit tools. In previous research [3], it took at least one week to teach students the skills they needed to complete a project that involved programming. Our new tools enabled us to introduce programming concepts much more quickly and intuitively.

8. DISCUSSION & FUTURE WORK

In light of the positive results from our workshops, we are now working to translate the curriculum content and principles presented here into a published book. In writing the curriculum, we continue to emphasize the values of learning by doing, encouraging early successes, and supporting multiple learning styles. We are careful in our writing to strike a balance between delving into technical details and teaching concepts only in contexts in which they are needed, in order to maintain student interest. To this end, we have added several projects and a programming and electronics reference section to our curriculum in order to avoid including overwhelming amounts of technical material in any single project, while still providing support for students who are interested in additional explorations.

We realize the difficulties inherent in translating our research into a curriculum that can be feasibly implemented in the real world by educators who are less experienced with the LilyPad Arduino. As

we compile our final materials, we are striving to be thoughtful about the constraints that educators face. For example, 1) educators are often restricted to teaching lessons in 1-2 hour chunks; 2) they may be teaching students with different levels of experience; 3) they may be learning programming and electronics themselves; and 4) they may have difficulty locating and acquiring all the materials necessary for a project. To address these concerns, we 1) provide time estimates for each activity; 2) have structured our projects to consist of a basic project with optional extensions; 3) write the book to speak directly to the end-user, be it student or educator; and 4) provide a materials reference section with detailed descriptions on what each object is, and where to acquire it. We have also created a LilyPad Arduino board that snaps on and off projects and is therefore reusable to mitigate cost concerns.

We plan to publish the curriculum as a self-contained book in 2013; we also intend to provide the full curriculum materials and additional support for lessons online. In doing so, we aim to maximize the resources available for educators in the real world to teach computer science using computational textiles.

9. ACKNOWLEDGMENTS

We wish to thank all of the students and educators who participated in our workshops. We would also like to thank Pete Lewis from SparkFun Electronics for his collaboration on developing the LilyPad ProtoSnap board.

10. REFERENCES

1. Baafi, E. Drag-n-Drop Arduino Programming. *MAKE Magazine 25*, 2011. http://makezine.com/25/modkit/.
2. Buechley, L., Eisenberg, M., Catchen, J., and Crockett, A. The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ACM (2008), 423–432.
3. Buechley, L., Eisenberg, M., and Elumeze, N. Towards a curriculum for electronic textiles in the high school classroom. *SIGCSE Bull. 39*, 3 (2007), 28–32.
4. Buechley, L. and Hill, B.M. LilyPad in the wild: how hardware's long tail is supporting new engineering and design communities. *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, (2010), 199–207.
5. Campbell, J., Gries, P., and Montojo, J. *Practical Programming: An Introduction to Computer Science Using Python*. Pragmatic Bookshelf, 2009.
6. Cooper, S., Dann, W., and Pausch, R. Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges 15*, 5 (2000), 107 – 116.
7. Dittert, N. and Schelhowe, H. TechSportiv: using a smart textile toolkit to approach young people's physical education. ACM (2010), 186–189.
8. Fields, D., Kafai, Y., and Searle, K. Functional Aesthetics for Learning: Creative Tensions in Youth e-Textile Designs. *Proceedings of the 10th International Conference of the Learning Sciences (ICLS 2012)*, (2012), 196–203.
9. Gibbons, M.T. The year in numbers. *Profiles of Engineering and Engineering Technology Colleges*, (2010), 11–46.
10. Gonzalez, N., Moll, L.C., and Amanti, C., eds. *Funds of Knowledge: Theorizing Practices in Households and Classrooms*. Lawrence Erlbaum Associates, 2005.
11. Gürer, D. and Camp, T. An ACM-W literature review on women in computing. *SIGCSE Bull. 34*, 2 (2002), 121–127.
12. Kafai, Y.B., Fields, D.A., and Searle, K.A. Making Learning Visible: Connecting Crafts, Circuitry & Coding in E-Textile Designs. *The Future of Learning: Proceedings of the 10th International Conference of the Learning Sciences (ICLS 2012), Volume 1, Full Papers*, (2012), 188–195.
13. Katterfeldt, E.-S., Dittert, N., and Schelhowe, H. EduWear: smart textiles as ways of relating computing technology to everyday life. *Proceedings of the 8th International Conference on Interaction Design and Children*, ACM (2009), 9–17.
14. Kelleher, C., Pausch, R., and Kiesler, S. Storytelling alice motivates middle school girls to learn computer programming. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2007), 1455–1464.
15. Kiesler, S., Sproull, L., and Eccles, J.S. Pool halls, chips, and war games: women in the culture of computing. *ACM SIGCSE Bulletin 34*, 2 (2002), 159–164.
16. Ladson-Billings, G. Toward a Theory of Culturally Relevant Pedagogy. *American Educational Research Journal 32*, 3 (1995), 465.
17. Lee, J.S. Technology education for woman by d.i.y. technology in closing gender gap. *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, ACM (2008), 3447–3452.
18. Lovell, E. and Buechley, L. LilyPond: an online community for sharing e-textile projects. *Proceedings of the 8th ACM conference on Creativity and cognition*, ACM (2011), 365–366.
19. Malan, D.J. and Leitner, H.H. Scratch for budding computer scientists. *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, ACM (2007), 223–227.
20. Millner, A. and Baafi, E. Modkit: blending and extending approachable platforms for creating computer programs and interactive objects. *Proceedings of the 10th International Conference on Interaction Design and Children*, ACM (2011), 250–253.
21. Ngai, G., Chan, S.C.F., Cheung, J.C.Y., and Lau, W.W.Y. The TeeBoard: an education-friendly construction platform for e-textiles and wearable computing. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2009), 249–258.
22. Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, 1980.
23. Turkle, S. and Papert, S. Epistemological Pluralism: Styles and Voices within the Computer Culture. *Signs: Journal of Women in Culture and Society 16*, 1 (1990), 128–57.
24. Zelle, J.M. *Python Programming: An Introduction to Computer Science*. Franklin Beedle & Associates, 2003.