Empowering Programmability for Tangibles

Eric Rosenbaum
Lifelong Kindergarten Group
MIT Media Lab
77 Massachusetts Ave.
Cambridge, MA 02139

Evelyn Eastmond
Lifelong Kindergarten Group
MIT Media Lab
77 Massachusetts Ave.
Cambridge, MA 02139

David Mellis
High-low Tech Group
MIT Media Lab
77 Massachusetts Ave.
Cambridge, MA 02139

TEI 2010, January 25-27, 2010, Cambridge, MA, USA
ACM

Abstract
Programming microcontrollers for tangible interfaces can be easier and more accessible than it is now, empowering a broader audience to participate. The first part of this studio will introduce participants to Scratch for Arduino, a graphical programming language for controlling the Arduino hardware platform. The participants will form small groups to create projects using the Arduino in combination with a kit of input and output devices, and program their creations' behavior using Scratch for Arduino. In the second part of the studio, participants will have a chance to get under the hood of the Scratch for Arduino language and its underlying blocks engine, modifying it or extending it to work with other tangible kits. We will close with a discussion about participants' experiences using and modifying Scratch for Arduino and the blocks engine, comparing them to other environments and considering possibilities for future work and collaborations.

Keywords
Graphical programming, empowerment, Arduino, Scratch

ACM Classification Keywords
K.3.1 Computer Uses in Education

Introduction

The group of people who are able to design and fabricate tangible interfaces is rapidly expanding, as the construction kits and design tools become more accessible. Hardware toolkits are rapidly appearing to fill a variety of niches. These include different combinations of low cost kits, open-source hardware, and specific audiences such as children, crafters, robotics hobbyists, and designers. Some examples are the Arduino [10], Lilypad [2], Phidgets [1], Makeboard [3], Gainer [4], LEGO Mindstorms [5], PicoCrickets [6], LogoChip [7], Sunspots [8], and D.tools [9]. The software environments for these kits are also diversifying, including a variety of programming languages, including wrappers for easier use of high level textual languages (such as the arduino programming environment), graphical environments (such as the software for LEGO Mindstorms, PicoCrickets, and LogoChip), and visual hardware design environments (such as Fritzing [11]).

In spite of this diversification, there is an open niche for a flexible toolkit with a programming environment that is both powerful and truly accessible to novices. Scratch for Arduino aims to fill this niche.

Scratch for Arduino is based on the Scratch programming environment [14]. Scratch allows people of all ages to create their own animations, games and interactive stories and share them on the web. In Scratch, you program by snapping together graphical blocks on the screen. The Scratch for Arduino environment uses the a very similar set of programming language metaphors as are used by Scratch, and they combine to make a powerful and flexible language that is easy for beginners to get started with.

A few other efforts are underway to use scratch-like programming to create an accessible programming environment (e.g. Modkit [12], Catenary [15], and Amici [13]). We see our work as complementary to these other efforts.

A graphical syntax that prevents errors

In the Scratch paradigm, each block represents a command that causes a character on the screen to do something like move, change color or make a sound. The graphical syntax illustrates which blocks can connect to each other and in what ways. For example, the blocks have tabs and slots at top and bottom, showing that they connect top to bottom to execute in sequence. They also have holders for arguments, which are rectangular for text, rounded for numbers, and hexagonal for true/false inputs. Blocks that output a value of a particular type have that shape and can only fit into a holder of that type. Because of these constraints, it is impossible to construct a program that is syntactically incorrect. Syntax errors, common in almost any other language, are frustrating and demotivating for beginners. By preventing these syntax errors altogether, we hope to create a much smoother learning process. Scratch for Arduino shares this same graphical syntax.

Tinkerability for real-time feedback

By tinkerability we refer to the ability to try out changes and see the results rapidly in real-time. While a Scratch program runs, you can modify it, re-arranging the blocks or changing parameters. There is an option to show the blocks highlighting as they

execture, and to view the value of any variable in real-time, so you can see exactly how your program is behaving as it runs. Scratch for Arduino has this same property of tinkerability.

Parallel processes
Programs in Scratch typically consist of several stacks of blocks which execute in parallel. This means there is an implicit use of multi-threading. Multi-threading is typically introduced as an advanced programming concept, but it aligns well with people's intuitions about how things in the world behave: objects, and especially creatures, have multiple behaviors and sub-systems that operate simultaneously. As a feature of Scratch it enables people to create complex behaviors easily. Parallel processes are unusual in microcontroller programming, especially for beginners, but this is mainly due to the nature of the programming environments that have been used. Scratch for Arduino takes of advantage of people's intuitions about parallel processes to enable them to create more complex behaviors for their creations.

Physical linked to virtual
In the Scratch for Arduino environment, the physical setup of the Arduino and the inputs and outputs connected to it can be linked to the representation on the screen. This link enables the programming to be simplified. The details of what is connected where are handled in a graphical configuration (for example, to show that an LED is connected to pin 7, the user would drag an icon of an LED to that pin on a representation of the Arduino). Rather than controlling settings for pins, the graphical blocks can express behaviors for particular inputs and outputs (for example, the block would say "LED on," and internally it would set pin 7).

A framework for blocks-based languages
Scratch for Arduino is just one instance of a wide range of possible graphical programming languages based on the Scratch paradigm. We are developing a framework based on Flash Actionscript 3.0 that allows people to easily create and customize their own blocks languages.

A path out of the sandbox
One of the major advantages of building this system on top of the existing Arduino platform is that we will be able to provide people a way to transition, as they become more advanced, away from Scratch for Arduino and into the more general purpose, lower-level platform, which is well-supported with a worldwide community.

Studio Pre-requisites

We will aim to accommodate a wide variety of skill and comfort levels with both programming and electronics. The Scratch for Arduino programming environment is designed for novices and should make it easy for anyone to get started, but it also has a high ceiling of complexity for those more experienced to explore. The Arduino electronics toolkit, with a set of easily pluggable inputs and outputs, should make it possible for beginners in electronics to get up to speed quickly, but the full range of electronics exploration will be available. In the second portion of the workshop, where participants will extend and modify the blocks language, more programming skill will be useful, but we expect that participants will work in groups and share expertise.

Studio Topics

The studio will be divided into two parts. Part one is exploring Scratch for Arduino, and part two is modifying and extending it. In part one, we will give an introduction to the Scratch for Arduino software system and the associated physical toolkit, consisting of the Arduino with a set of input and output devices. Participants will have hands-on time to experiment with the system, develop a simple project and present it to the group, along with the reflections on the use of the system. Part Two will involve a more detailed look at the programming environment, and an investigation of its underlying software. We will get the participants set up to program in the Adobe flex environment so they can try out modifying the Scratch for Arduino code. They can start with the Scratch blocks engine and connect it to another hardware or software toolkit of their choice. Finally, we close with a conversation in which participants reflect on their experiences using Scratch for Arduino and then extending and modifying it. We will talk about future directions, including possible collaborations on graphical programming environments, and more generally the research endeavor of creating more accessible and empowering programming environments for tangibles.

Studio Learning Goals

We expect participants to learn how to use the Scratch for Arduino software to program a tangible system, and begin the process of learning how to modify and extend it. We also expect participants to engage in a reflective conversation about the space of hardware and software toolkits for tangibles, their different properties as they relate to accessibility and empowerment, and future possibilities for new toolkits.

Studio Supporting Web Documents

We plan to create a website in advance of the workshops that participants can use to access the Scratch for Arduino software, and get set up to use and modify it using the Adobe Flex Builder environment.

References

1. Phidgets Inc. - Unique and Easy to Use USB Interfaces. http://www.phidgets.com/

2. Lilypad. http://www.arduino.cc/en/Main/ArduinoBoardLilyPad.

3. makezine.com: Controller Kit. http://makezine.com/controller/

4. GAINER. http://gainer.cc/

5. LEGO.com MINDSTORMS NXT Home. http://mindstorms.lego.com/

6. PicoCricket – Invention kit that integrates art and technology. http://www.picocricket.com/

7. LogoChip. http://www.wellesley.edu/Physics/Rberg/logochip/

8. SunSPOTWorld - Home. http://www.sunspotworld.com/

9. HCI at Stanford University: d.tools. http://hci.stanford.edu/research/dtools/

10. Arduino. http://www.arduino.cc/

11. Fritzing. http://fritzing.org/

12. Modkit. http://www.modk.it/

13. Amici. http://dimeb.informatik.uni-bremen.de/eduwear/?cat=4

14. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., and Eastmond, E. Scratch: Programming for Everyone. Communications of the ACM, 2009.

15. Catenary - Scratch Connections. http://scratchconnections.wik.is/User:Chalkmarrow/Catenary