Empowering Programmability for Tangibles

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

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

Copyright is held by the author/owner(s).
TEI 2010, January 25-27, 2010, Cambridge, MA, USA
ACM 978-1-60558-841-4/10/01.
General Terms
design

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, like 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. Arduino is an open-source hardware and software platform for microcontroller programming. It is available worldwide. 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 (see figure 1), and eventually transition to the existing easy-to-use textual programming environment.

A few other efforts are underway to use scratch-like programming to create an accessible programming environment for Arduino (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
The Scratch paradigm reduces frustration by preventing people from building programs that will not run. The syntactic constraints are enforced by graphical constraints, so you cannot connect blocks incorrectly.

Tinkerability for real-time feedback
By tinkerability we refer to the ability to try out changes and see the results rapidly in real-time. You can click on a block to run it immediately, see the values of variables changing in real time, and modify the program as it runs.

Parallel processes
Programs in Scratch can consist of several stacks of blocks which execute in parallel. This implicit use of multi-threading aligns with people’s intuitions about behaviors, enabling them to create more complex creations.

Figure 1: A program in Scratch for Arduino
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 that will allow people to easily create and customize their own blocks languages.

Studio Pre-requisites
We will aim to accommodate a wide variety of skill and comfort levels with both programming and electronics.

Studio Topics
The studio will be divided into two parts. In part one, participants will use Scratch for Arduino to create projects. In part two they will modify and extend Scratch for Arduino. We will close with reflections and time to document the activities of the workshop.

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