Programming Revisited – The Educational Value of Computer Programming

Eric Klopfer, Mitchel Resnick, John Maloney, Brian Silverman, Andrea diSessa,
Andrew Begel & Chris Hancock
77 Massachusetts Ave, MIT Bldg 7-337, Cambridge, MA 02139
Tel: 617-253-2025, Fax: 815-327-4178
Email: klopfer@mit.edu

Abstract: This panel will address pedagogical needs for revisiting the role of computer programming for student learning. We will explore advances in programming platforms that enable students to create compelling projects with new technologies, and discuss the affordances of these new initiatives. We will address how these tools and techniques can be integrated into the curriculum of the classroom as well as informal learning environments.

Panel Overview

"*Any sufficiently advanced technology is indistinguishable from magic*" Arthur C. Clarke

As technologies become more and more advanced, we have empowered students to magically create things themselves on computers. They can manipulate media, explore virtual worlds, and communicate across the globe almost effortlessly. The day when students need to program computers to do interesting things is far behind us. So what is the role of computer programming for students at this point in time? What are the advantages to kids of understanding and controlling the science behind the magic?

This panel will address pedagogical needs for revisiting the role of computer programming for student learning. We will explore advances in programming platforms that enable students to create compelling projects with new technologies, and discuss the affordances of these new initiatives. We will address how these tools and techniques can be integrated into the curriculum of the classroom as well as informal learning environments.

The panel will consist of leaders in the development and study of programming tools, specifically for secondary school aged students including

- Mitchel Resnick, Brian Silverman, and John Maloney- MIT
- Andrea diSessa- UC Berkeley
- Eric Klopfer and Andrew Begel- MIT/UC Berkeley
- Chris Hancock - Tertl Studos

Scratch: Creating a "Programming Culture" at Community Technology Centers

Mitchel Resnick, John Maloney, Brian Silverman
MIT Media Laboratory

Walk into a Computer Clubhouse (part of a network of after-school centers in low-income communities) and you are likely to see youth creating and manipulating graphics, animations, videos, and music (and often integrating multiple media). A "Photoshop culture" has emerged at many Clubhouses, with youth proudly displaying their Photoshop creations on bulletin boards (both physical and online), sharing Photoshop techniques and ideas with one another, and helping Clubhouse newcomers get started with the software. But youth at Computer Clubhouses rarely become engaged in computer programming. Part of the problem, we believe, is that computer programming has been introduced using programming languages that are difficult to use, with proposed activities that are not deeply connected to young people's interests and passions.

We are developing a new programming environment, called Scratch, as part of an effort to foster a "programming culture" analogous to the "Photoshop culture" in Computer Clubhouses. Scratch aims to build on the interests of Clubhouse youth, adding programmability to the media-rich and network-based activities that are

most popular in Clubhouses. In this session, we will discuss the design principles and core features of the Scratch environment (drag-and-drop programming, programmable manipulation of rich media, shareability of Scratch objects, seamless integration with the physical world). More broadly, we will discuss what we mean by a "programming culture," why we feel it is important, how we are trying to foster it, and the challenges that stand in the way.

Boxer

Andrea diSessa
University of California, Berkeley

Andrea diSessa has long advocated a "literacy" model for computer use--that is, everyone should know how to produce (as well as "consume") dynamic, interactive representations, just as everyone in our society learns to read. One of the counter-intuitive implications of this point of view is that we should be striving to produce the most general-purpose and flexible platforms possible--like written text--not "a separate application for every purpose."

For this panel, diSessa proposes to stress two ideas. First, the intellectual power that programming representations can have for learning science is at least comparable to, if not far greater than, algebra. Hence "avoiding programming" would be like "avoiding equations" in science learning. The second idea has to do with minimizing programming in using computational media when it is NOT intellectually important. New techniques are needed, and are in development, to make the set of things that can be done "within epsilon of not programming" much bigger. One important strategy is to populate computational media with flexible, modifiable, and interoperable tools (components) that bring specialized functionality within easy reach without subverting the intrinsic flexibility of computational media. Realizing the promise of components, however, is trickier than most expect.

StarLogo TNG

Eric Klopfer & Andrew Begel
MIT Teacher Education Program and University of California, Berkeley

StarLogo is a language designed to facilitate the creation of complex systems models by students and teachers. Constantly evolving, it has been widely used by secondary school students and teachers. We have been working intensively with teachers over the last several years to help them integrate StarLogo into their curriculum. The problem that we have encountered is that many science and math teachers use the models that they build, but don't feel they have the time or expertise to teach students to build models in their own classes. Yet, they believe that students will develop a deeper understanding if they could just have that opportunity. In response to this we have created StarLogo TNG, a graphical programming environment that takes a new perspective on StarLogo model creation and representation.

StarLogo TNG is a block-based graphical programming language. While drag and drop programming itself is not new, we have introduced features that make it easier for novices to begin programming while maintaining the depth and flexibility that more sophisticated programmers require. For example, fixed-size blocks offering the user only limited visual feedback have been replaced by blocks that dynamically change shape to illustrate when a programmer tries to perform a gestural action that is incorrect. Animations also help programmers add parameters to user-defined procedures in a fluid, intuitive way.

In addition to making construction of programs easier, we have also introduced a 3D perspective into StarLogo that will help people conceive and understand their models. Evoking the metaphor of our Participatory Simulations (that put people in the role of a "turtle" as they engage in interactive real world activities), our 3D "turtles-eye" view of the world is intended to similarly help people bridge their real-world understanding with the models that they create in StarLogo.

Early feedback suggests that these new features will help teachers integrate programming and model building into their curriculum.

Real-Time Models and their Educational Significance

Chris Hancock
Tertl Studos

Several writers have observed that programming can be a kind of modeling activity, in which the requirements of organizing a working program lead children into engagement with a process of situated abstraction (Eisenberg, 1995, Kynigos, 1995, Noss & Hoyles, 1996). This presentation takes that idea a step further, to the idea of a live, real-time model. Real-time domains such as robotics are especially appealing to young programmers, but also pose special programming difficulties. Work on the Flogo I and Flogo II programming environments aims to make real-time programming more accessible to learners, in part through the development of "live" environments in which programs reveal their own execution as they run, and can be modified in mid-execution. I will present some simple examples of real-time modeling as exhibited by young learners of robotics programming, and discuss the educational significance of programming as a context and medium for live modeling of dynamic phenomena.

References

Eisenberg, M. (1995). Creating software applications for children: Some thoughts about design. In diSessa, A., Hoyles, C., and Noss, R. (Eds.) *Computers and Exploratory Learning*. NATO ASI Series F. Springer. Berlin.

Kynigos, C. (1995). Programming as a means of expressing ideas: Three case studies situated in a directive educational system. In diSessa, A., Hoyles, C., and Noss, R. (Eds.) *Computers and Exploratory Learning*. NATO ASI Series F. Springer. Berlin.

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers*. Dordrecht the Netherlands: Kluwer Academic Publishers.