Déjà Vu?

Michael Tempel
Logo Foundation
michaelt@media.mit.edu

Abstract
In the early 1980s computers began to enter schools, many running Logo. The constructivist roots of Logo did not prevent its being used in instructionist ways, while at the same time, most progressive educators were not attracted to it. In recent years we have seen a rising interest in teaching computer programming. Is there a similar pattern now, with constructionist learning environments, notably Scratch, being used in instructionist ways? And where are the progressive educators today?

Keywords constructionism, instructionism, progressive education, computer programming, coding

Then

As a starting point let's take the 1971 paper by Seymour Papert and Cynthia Solomon: *Twenty Things to Do with a Computer*[1]. Then, fast forward to Mitch Resnick's 2012 article in *Educational Technology*: "Reviving Papert's Dream"[2] where he refers to the Papert and Solomon paper as setting the model for what children can do with computers and what is currently being pursued with Scratch.

Now let's look at two time periods: the early 1980s, when microcomputers began to go into schools in increasing numbers, and the past few years. Similarities emerge. In the 1980s Logo was adopted by some of the people advocating that everyone should learn to program. In that context the discussion was about how to teach programming and should we use BASIC, Logo, or Pascal. Rather than Logo being an environment for exploring, creating, and learning, it became the subject to be taught.

At the same time, progressive educators for the most part were not interested in using computers, even with Logo. The early microcomputers found in schools were clumsy and difficult to use. The range of possible activities was seen as narrow. Most of the Twenty Things to Do with a Computer were difficult or impossible.

Persistent Themes

Recently we have been seeing something similar. Again, we are hearing that everyone should learn to program, although now it's called "coding." Programming environments - most notably Scratch -

[1] ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-248.pdf
[2] http://web.media.mit.edu/~mres/papers/educational-technology-2012.pdf

that are designed to support constructionist learning are also being adopted by people with an instructionist mindset. So what is different and what is the same between then and now?

What is constant is the dominance of Instructionism in educational systems worldwide, both in the institutions and in people's thinking. The Instructionist meme is deeply ingrained and widely accepted without much reflection or questioning.

An interesting variant of this pattern, which we often saw with Logo, can be summed up as "Constructionism for me, instruction for you." A teacher would come to a Logo workshop and have a wonderful experience learning and creating, but then feel the need to create a curriculum and lesson plans to teach his students what he had just learned. We see this happening with contemporary programming environments as well.

This phenomenon is partly driven by teachers' enthusiasm about learning. A good way to learn is to teach. By "teaching to learn" they are instructing as part of the process of constructing their own knowledge. It takes restraint, skill, and planning to create an environment for students to have that same experience.

Now

What is different today? All of the Twenty Things to Do with a Computer, and much more, are now possible. The appeal of computing is much broader than it was in the early 1980s. There are many more possibilities for construction of artifacts and projects that many more people care about.

Another difference is that access to computing is now pervasive, and related to that, the fact that computers now do a great many things that people enjoy in their day-to-day lives. This context provides models for what one can create, and then the opportunity to create apps and devices that really are useful to one's self and to others.

Programming environments and their communities of users are no longer separate. A good example is Scratch, where the website includes the programming environment, millions of projects and a large community of Scratchers. This access to people and resources supports Constructionism and undermines Instructionism.

Even when web-based resources are Instructionist in intent, shifting control of access to the user supports Constructionism. If I choose to watch your online lecture, or part of it, when I need it to support a project I'm working on, it may be Instructionism for you, but it's Constructionism for me.

But will all of this simply become curricularized as it is taken up by schools? The question is whether Constructionism now has a friendlier environment within which to live and grow, or will the Instructionist meme continue to dominate the educational culture.

In *Mindstorms* Seymour Papert presents a vision of learning in a computer culture for 176 pages without discussing school. Then, in the final chapter he addresses the issue and finds a closer kinship to informal learning environments than to school. But he also points to an important difference between the early Logo environments and the samba school: Logo was separate from the surrounding culture while the samba schools were rooted in it. The most significant difference between the early 1980s and now is that computing is increasingly enmeshed in the broader culture.

How schools will respond to this change remains an open question. Maybe this time around progressive educators will become interested.