History and Programming

Eduardo Bellani *

November 24, 2010

Abstract

Those who cannot remember the past are condemned to repeat it. Because programmers usually think they deal with cutting edge technology, they tend to forget the age and genealogy of the ideas they are working with. A demonstration of the history of the some crucial ideas of the programming craft would avoid the repetition of error and allow better ideas to take hold.

keywords: Programming, History, Learning

1 Introduction

Almost every school that tries to form professionals(?) begins with an overview of the history of the field. That is doubly true (?) in art schools, because one is expected to learn to develop his own style by studying the style of others. Programming, which is usually seen as an art discipline as much as an engineering one, does not maintain that tradition. That leads to a lack of vision an historical perspective of the discipline methods, points of views and techniques.

A professional without an historical perspective could be compared to a navigator without a compass. The navigator can move, but hardly advance towards a goal if not by accident.

2 Ideas

Here I will approach three of the main ideas that integrated to form the modern personal computer environment: Graphical user interface, constructionism, and collaborative systems.

*blog – contact

This division will at times seem artificial, not only because they influenced each other, but because there is great overlapping of techniques and philosophy between them. On the other hand this structure does capture important distinctions and points of view that are valuable to understand the whole story.

2.1 GUI

The development of all graphic environments can be traced directly to Ivan Sutherland's sketchpad. Its ideas *still influence how every computer user thinks about computing.*Sutherland [2003]

It did that by introducing graphic metaphors and devices, such as the light-pen, the rubber band graphic manipulation, icons and a great deal of the things most people take as "natural" when manipulating modern GUI. Sutherland [2003]

But the major influence for programming paradigms was because its object oriented features, which along with Simula [1] had a major impact in the inspiration and development of Smalltalk:

> What Simula was allocating were structures very much like the instances of Sketchpad. There were descriptions that acted like masters and they could create instances, each of which was an independent entity. What Sketchpad called masters and instances, Simula called activities and processes. ...
>
> This was the big hit, and I've not been the same since. Kay [1993]

Figure 1: Sketchpad console, circa 1962. Müller-Prove [2002]

2.2 Constructionism

Programming is an activity that deals primarily with the construction of abstract structures from ideas. This process and some of its encompassing theories are usually neglected or just fade into the conceptual background. This void is filled by constructionism, a learning theory developed by Seymour Papert and his colleagues that:

[1] Both share a common ancestor in the work of Douglas T. Ross Sutherland [2003]

> ... shares constructivism's connotation of learning as "building knowledge structures" irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe. Papert and Harel [1991]

The proponents of this theory were the front-runners of using computers as a *personal media*. They did that in order to propose an alternative to instructionism and technocentrism. Those ideas can be summarized in the claim that *"To get better education, we must improve instruction. And if we're going to use computers, we'll make the computers do the instructon."*. Papert [1980]

Constructionism's alternative consists of using media, technology, and social environments as scaffolds to the conceptual structures that is to be built by people in the active role of their education. This education was initially children learning math supported by constructionism's answer to traditional math curricula, Logo.

Logo profoundly influenced modern computer languages and environments through the heavy impact it had on the development of the Smalltalk system:

Figure 2: Children programming in logo. Papert [1981]

> I finally visited Seymour Papert, Wally Feurzig, Cynthia Solomon and some of the other original researchers who had built LOGO and were using it with children in the Lexington schools. Here were children doing real programming with a specially designed language and environment. As with Simula's leading to OOP, this encounter finally hit me with what the destiny of personal computing really was going to be. Not a personal dynamic vehicle, as in Engelbart's metaphor opposed to the IBM "railroads", but something much more profound: a personal dynamic medium. With a vehicle one could wait until high school and give "drivers ed", but if it was a medium, it had to extend into the world of childhood. Kay [1993]

References

Alan C. Kay. The early history of Smalltalk. *SIGPLAN Not.*, 28(3):69–95, March 1993. ISSN 0362-1340. doi: 10.1145/155360.155364. URL http://dx.doi.org/10.1145/155360.155364.

M. Müller-Prove. *Vision and Reality of Hypertext and Graphical User Interfaces*. PhD thesis, University of Hamburg, February 2002.

Seymour Papert. Constructionism vs. instructionism. 1980. URL http://www.papert.org/articles/const_inst/const_inst1.html.

Seymour Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, January 1981. ISBN 0465046274.

Seymour Papert and Idit Harel. *Situating Constructionism*. Ablex Publishing, 1991. URL http://www.papert.org/articles/SituatingConstructionism.html.

Ivan Edward Sutherland. Sketchpad: A man-machine graphical communication system. 2003. URL http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-574.pdf.