An interview with Lev Manovich about his book **Software takes command**
6 May 2014

Software Takes Command

An Interview with Lev Manovich

MICHAEL CONNOR | Wed Jul 10th, 2013 9 a.m.

Lev Manovich is a leading theorist of cultural objects produced with digital technology, perhaps best known for The Language of New Media *(MIT Press, 2001). I interviewed him about his most recent book,* Software Takes Command *(Bloomsbury Academic, July 2014).*

MICHAEL CONNOR: I want to start with the question of methodology. How does one study software? In other words, what is the object of study—do you focus more on the interface, or the underlying code, or some combination of the two?

LEV MANOVICH: The goal of my book is to understand media software—its genealogy (where does it come from), its anatomy (the key features shared by all media viewing and editing software), and its effects in the world (pragmatics). Specifically, I am concerned with two kinds of effects:

1) How media design software shapes the media being created, making some design choices seem natural and easy to execute, while hiding other design possibilities;

2) How media viewing / managing / remixing software shapes our experience of media and the actions we perform on it.

I devote significant space to the analysis of After Effects, Photoshop and Google Earth—these are my primary case studies

I also want to understand what media is today conceptually, after its "softwarization." Do the concepts of media developed to account for industrial-era technologies, from photography to video, still apply to media that is designed and experienced with software? Do they need to be

Frieder Nake
nake@informatik.uni-bremen.de | (0421) 218 64485 | Linzer Str. 9a, room 3015 | www.agis.informatik.uni-bremen.de |
@CarlCanary | Office hours Thursday 14 to 16 at Linzer Str. 9a | or Monday upon personal arrangement at HfK

updated, or completely replaced by new more appropriate concepts? For example: do we still have different media or did they merge into a single new meta-medium? Are there some structural features which motion graphics, graphic designs, web sites, product designs, buildings, and video games all share, since they are all designed with software?

In short: does "media" still exist?

For me, "software studies" is about asking such broad questions, as opposed to only focusing on code or interface. Our world, media, economy, and social relations all run on software. So any investigation of code, software architectures, or interfaces is only valuable if it helps us to understand how these technologies are reshaping societies and individuals, and our imaginations.

MC: In order to ask these questions, your book begins by delving into some early ideas from the 1960s and 1970s that had a profound influence on later developers. In looking at these historical precedents, to what extent were you able to engage with the original software or documentation thereof? And to what extent were you relying on written texts by these early figures?

LM: In my book I only discuss the ideas of a few of the most important people, and for this, I could find enough sources. I focused on the theoretical ideas from the 1960s and 1970s which led to the development of modern media authoring environment, and the common features of their interfaces. My primary documents were published articles by J. C. R. Licklider, Ivan Sutherland, Ted Nelson, Douglas Engelbart, Alan Kay, and their collaborators, and also a few surviving film clips—Sutherland demonstrating Sketchpad (the first interactive drawing system seen by the public), a tour of Xerox Alto, etc. I also consulted manuals for a few early systems which are available online.

While I was doing this research, I was shocked to realize how little visual documentation of the key systems and software (Sketchpad, Xerox Parc's Alto, first paint programs from late 1960s and 1970s) exists. We have original articles published about these systems with small black-and-white illustrations, and just a few low resolution film clips. And nothing else. None of the historically important systems exist in emulation, so you can't get a feeling of what it was like to use them.

This situation is quite different with other media technologies. You can go to a film museum and experience the real Panoroma from early 1840s, camera obscura, or another pre-cinematic technology. Painters today use the same "new media" as Impressionists in the 1870s—paints in tubes. With computer systems, most of the ideas behind contemporary media software come directly from the 1960s and 1970s—but the original systems are not accessible. Given the number of artists and programmers working today in "software art" and "creative coding," it should be possible to create emulations of at least a few most fundamental early systems. It's good to take care of your parents!

MC: One of the key early examples in your book is Alan Kay's concept of the "Dynabook," which posited the computer as "personal dynamic media" which could be used by all. These ideas were spelled out in his writing, and brought to some fruition in the Xerox Alto computer. I'd like to ask you about the documentation of these systems that does survive. What importance can we attach to these images of users, interfaces and the cultural objects produced with these systems?

LM: The most informative sets of images of Alan Kay's "Dynabook" (Xerox Alto) appears in the article he wrote with his collaborator Adele Goldberg in 1977. In my book I analyze this article in detail, interpreting it as "media theory" (as opposed to just documentation of the system). Kay said that reading McLuhan convinced him that computer can be a medium for personal expression. The article presents theoretical development of this idea and reports on its practical implementation (Xerox Alto).

Alan Turing theoretically defined a computer as a machine that can simulate a very large class of other machines, and it is this simulation ability that is largely responsible for the proliferation of computers in modern society. But it was only Kay and his generation that extended the idea of simulation to media—thus turning the Universal Turing Machine into a Universal Media Machine, so to speak. Accordingly, Kay and Goldberg write in the article: "In a very real sense, simulation is the central notion of the Dynabook." However, as I suggest in the book, simulating existing media become a chance to extend and add new functions. Kay and Goldberg themselves are clear about this—here is, for example, what they say about an electronic book: "It need not be treated as a simulated paper book since this is a new medium with new properties. A dynamic search may be made for a particular context. The non-sequential nature of the file medium and the use of dynamic manipulation allow a story to have many accessible points of view."

The many images of media software developed both by Xerox team and other Alto users which appear in the article illustrate these ideas. Kay and Goldberg strategically give us examples of how their "interim 'Dynabook'" can allow users to paint, draw, animate, compose music, and compose text. This maked Alto first Universal Media Machine—the first computer offering ability to compose and create cultural experiences and artifacts for all senses.

MC: I'm a bit surprised to hear you say the words "just documentation!" In the case of Kay, his theoretical argument was perhaps more important than any single prototype. But, in general, one of the things I find compelling about your approach is your analysis of specific elements of interfaces and computer operations. So when you use the example of Ivan Sutherland's Sketchpad, wasn't it the documentation (the demo for a television show produced by MIT in 1964) that allowed you to make the argument that even this early software wasn't merely a simulation of drawing, but a partial reinvention of it?

LM: The reason I said "just documentation" is that normally people dont think about Sutherland, Engelbart or Kay as "media theorists," and I think it's more common to read their work as technical reports.

On to to Sutherland. Sutherland describes the new features of his system in his Ph.D. thesis and the published article, so in principle you can just read them and get these ideas. But at the same time, the short film clip which demonstrates the Sketchpad is invaluable—it helps you to better understand how these new features (such as "contraints satisfaction") actually worked, and also to "experience" them emotionally. Since I have seen the film clip years before I looked at Sutherland's PhD thesis (now available online), I can't really say what was more important. Maybe it was not even the original film clip, but its use in one of Alan Kay's lectures. In the lecture Alan Kay shows the clip, and explains how important these new features were.

MC: The Sketchpad demo does have a visceral impact. You began this interview by asking, "does media still exist?" Along these lines, the Sutherland clip raises the question of whether drawing, for one, still exists. The implications of this seem pretty enormous. Now that you have established the principle that all media are contingent on the software that produces, do we need to begin analyzing all media (film, drawing or photography) from the point of view of software studies? Where might that lead?

LM: The answer which I arrive to the question "does media still exist?" after 200 pages is relevant to all media which is designed or accessed with software tools. What we identify by conceptual inertia as "properties" of different mediums are actually the properties of media software—their interfaces, the tools, and the techniques they make possible for navigating, creating, editing, and sharing media documents. For example, the ability to automatically switch between different views of a document in Acrobat Reader or Microsoft Word is not a property of “text documents,” but as a result of software techniques whose heritage can be traced to Engelbart’s “view control.” Similarly, "zoom" or "pan" is not exclusive to digital images or texts or 3D scenes—its the properly of all modern media software.

Along with these and a number of other "media-independent" techniques (such as "search") which are build into all media software, there are also "media-specific" techniques which can only be used with particular data types. For example, we can extrude a 2-D shape to make a 3D model, but we can't extrude a text. Or, we can change contrast and saturation on a photo, but these operations do not make sense in relation to 3D models, texts, or sound.

So when we think of photography, film or any other medium, we can think of it as a combination of "media-independent" techniques which it shares with all other mediums, and also techniques which are specific to it.

MC: I'd proposed the title, "Don't Study Media, Study Software" for this article. But it sounds like you are taking a more balanced view?

LM: Your title makes me nervous, because some people are likely to misinterpret it. I prefer to study software such as Twitter, Facebook, Instagram, Photoshop, After Effects, game engines, etc., and use this understanding in interpreting the content created with this software—tweets, messages, social media photos, professional designs, video games, etc. For example, just this morning I was looking at a presentation by one of Twitter's engineers about the service, and learned that sometimes the responses to tweets can arrive before the tweet itself. This is important to know if we are to analyze the content of Twitter communication between people, for example.

Today, all cultural forms which require a user to click even once on their device to access and/or participate run on software. We can’t ignore technology any longer. In short: ”Software takes command.“

Note. *The title of a famous book by Sigfried Giedion was ”Mechanization takes command.“ It was publkished first in 1948*