

## Speaking Computer



# Sometimes how you design is as important as what you design.

I received a letter from a human-rights group the other day and, even before opening the envelope, I knew a friend of mine had designed the stationery. There were very few clues to go on—actually, just the return address—but something about the choice of fonts and colors and the way the words related to one another tipped me off. I recognized the unique impression her style and personality had made upon her work.

This isn't uncommon. If you've ever been in a class where the teacher held up an example of a design or artwork and you knew, before he told you, which of your classmates had made it, then you know what I mean. Who we are shapes what we make.

Dutch social anthropologist Dr. Geerte Hofstede has studied this phenomenon on a cultural, as well as a personal, level. "The influence of culture is so powerful that one can almost always, when reading a book for instance, recognize the nationality of the author, even if it has not been mentioned," Hofstede writes. "This applies to our work too—we are from Holland, and even when we write in English, the Dutch software of our minds will remain evident to the careful reader."<sup>1</sup>

Hofstede has developed a system for understanding how the *software of our minds* affects and even determines what we make. His theory attempts to explain why, for instance, Italians design beautiful cars and shoes and Americans were the first to put a rocket on the moon.

It's no coincidence that Hofstede compares our mental processes to software. Just like our personalities, the software we use leaves its mark on our design. Whether we choose to hand-code a Web site in HTML or use Dreamweaver will make a difference in the final result. Hand-coding, for example, allows us to focus on the details and make every page unique, while *wysiwyg* (what you see is what you get) programs such as Dreamweaver encourage templating. How you make something determines what you make.

But we're not just talking about Web sites here. Tools also leave their mark on print projects, motion graphics, and product design. One of the reasons that these traces are sometimes difficult to detect is that, at the moment, almost every designer in the world is using the same software made by the same companies. There just aren't very many cultures to compare. This may be changing, as more and more designers begin to create and modify their own software.

<sup>1</sup>Hofstede, *Geert. Cultures and Organizations, Software of the Mind*. McGraw-Hill, 1996.

But regardless of what software you use or whether you make Web sites, running shoes, or movies, there is a language that we all have in common, and that is the language of the computer.

The good news is that to speak computer you have to learn only two words. But those two words make all the difference.

## Yes and No

At its core, the binary language of the computer is very simple. In fact, it has only two words: yes and no, which are represented as one or zero. It's just like being in court. Please answer the question: Did you commit the crime? Yes or no? The prosecutor does not want to hear about who said what to whom first or whether you skipped breakfast. In the same way the language of the courtroom affects the final verdict, the language of the computer affects what is produced and how.

The binary language of computers has led to some amazing efficiencies, perhaps the most basic being that almost no one has to actually input ones and zeros. The digits have been clumped together into words (which are a collection of six or eight numbers or bits), and the words have been structured into languages. The languages generate visual interfaces that allow you to draw instead of type. So, rather than playing twenty questions every time you want to make a shape—say a circle—you can simply draw the circle and the computer will translate the shape into ones and zeros for you. And, because the computer is using an algorithm to describe the shape, the form is perfectly precise. The computer can also replicate forms perfectly. If you want to make a hundred or a thousand or a million circles of the same size or of all different sizes, all you have to do is ask.

“Computer programming is just like speaking a language,” digital designer Jonathan Puckey told me. “As long as I can describe in words what I want to accomplish, I can put those ideas into code.”

The ability of the computer to create precise forms and to replicate forms exactly has had an enormous impact on the way things that surround us look and behave. The revolution brought about by computers can be seen as a continuation of the revolution that began with the assembly line. Each of the Model T cars coming off the assembly line used the same parts, and each car looked identical (or very close to identical).

What Ford did to production, digital technology has done to design. By automating the design process, we can work much more rapidly, and with greater precision. Of course, as in the case of the Model T, we might eventually get tired of everything looking more or less similar.

And this is where things start to get interesting. You can ask the computer to draw a perfect circle, but you can also ask it to draw an imperfect circle, or a thousand circles, each of which is imperfect in a different way. Video game designers do this all the time—it's the reason why, as you plow through the bushes in *Grand Theft Auto*, each leaf on each bush looks realistic: By *realistic* we mean that they are all slightly different though more or less the same.

But the imperfections don't need to be slight. I often hear digital designers say that one of the things they really love about computers is that computers allow them to screw things up fast. One small change in the code—a one here, a zero there—and things can get really weird. Digital designers, even those who really know their code, often can't guess what the results of their experiments will be. But that's the fun of it. And, once in a while, something unexpected will catch their eye and cause them to see a shape or line in a different way and they will play around a little more. That is how new tools and new ideas in design are born.

But, lest you think that all experiments are successful, the language of ones and zeros can also cause trouble, as you will already know if you've ever had your computer crash. There are lots of different reasons for computer crashes, but here's one way to make software failure almost inevitable.

Let's say you are working on a project and the deadline is approaching. You are writing code, and suddenly you remember that your partner did a project kind of like this a few months back. So you pick up the section of code that approximates the effect or function you are looking for. You make a few modifications. Although you know it's not the best way to solve the problem, you really only care about one question: Does it work? Yes or no. You get a yes, and you move on. A few weeks later, your partner is working on another project, and this time she picks up the code you just wrote and makes more modifications.

Does it still work? Well, the real answer is barely, but in code, as in the courtroom, there is no room for nuance. So the answer goes down as yes. Over time, all of these yes answers pile up as the code gets passed around, and each time it gets uglier and less efficient. Finally, you ask, "Does it work?" and the answer is a resounding *no*. Because so many replications and modifications have been made to the program in the interest of saving time, trying to figure out exactly what went wrong is like untangling a really awful knot. In fact, sometimes it's easier to scrap the whole thing and start over.

## LeWitticisms

In the late 1960s, the artist Sol LeWitt decided that the thinking behind a piece of art is just as interesting as the final result. So, he put down his paintbrush and started issuing instructions. Here are his instructions for Wall Drawing #46: "Vertical lines, not straight, not touching, covering the wall evenly."

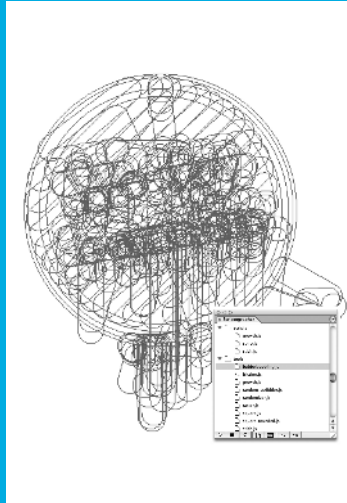
That's it. Then it was up to the galleries to follow his instructions and create the actual piece, usually by drawing with pencils directly on the gallery wall. By allowing for the separation of insight and execution, LeWitt anticipated the digital age.

Like LeWitt, digital designers are often in a position of issuing instructions, which are then interpreted by computers and the people who use them. The fact that no two galleries executed his instructions in exactly the same way did not bother LeWitt—in fact, the variation is part of what makes the artwork interesting.

Computers don't always interpret instructions in exactly the same way, either. The instructions, or lines of code, will yield different results when they are interpreted by NASA's supercomputers and a cell phone. Likewise, many of the most successful online experiments rely on millions of individual users to interpret instructions. For example, eBay, with its plaster figurines and paintings of ponies, isn't going to win any beauty contests, but what makes it so effective is the simplicity of the instructions. In effect, eBay says, "Post your treasures online and let people bid on them." When you are evaluating digital design, it's important to understand not only what something looks like, but also the instructions that led to its creation.

## Free Software (Your Design Will Follow)

Although commercial software dominates digital design, there are enough free tools available to complete a project if you are willing to be creative. The advantages of using free software—other than that it's free—are that it forces you to take a detour from the mainstream. By using different tools, you are likely to get different results. You are also likely to learn more about your tools because you will be part of the process of developing them.



▲ **Scriptographer** ©2000 - 2007,  
Jürg Lehni

Libre Graphics is an international organization devoted to developing and using free graphics software. The organization supports the development of tools such as Gimp, an image-manipulation program; Inkscape, a vector-drawing package similar to Adobe Illustrator; and Scribus, a page-layout program.

Recently, the idea of free tools has been getting some major mainstream support. Google has been adding free tools to its repertoire. In addition to photo, document, and spreadsheet software, it also has a free 3-D software tool called Sketch Up that allows you to create elaborate models of anything from cities to software systems. Meanwhile, over at MIT Media Lab, John Maeda is working on a project called OpenStudio, which currently includes free drawing tools but may be expanded and developed to allow collaboration in multiple media.

If you're not ready to "go open" all at once, there are groups of designers creating free plug-ins that extend the functionality of commercial software. One of my favorites is Scriptographer.com, which creates new tools that run on top of Adobe Illustrator.

## Language and Community

In the early 1900s, hundreds of Native American children were sent off to boarding schools run by missionaries so that they could receive a Christian education. One of the rules the children had to follow was that they must not speak their tribal language. Even after they were fluent in English, the children, now old, recall sneaking out to have whispered conversations in their native tongues.

From one perspective, this seems pretty odd—after all, isn't what they were saying more important than the language it was said in? But the children recognized that their languages provided a connection to their heritage and home, a way to identify one another and distinguish themselves from white society. When speaking their own languages, they once again felt part of the tribe.

In the frontier landscape of digital technology, language also gives people something to rally around and can serve as a means of protest. What is now called the *open-source movement* was an attempt to establish a programming language that was not dominated by a single corporation. The *source* in open source refers to source code. The idea is that anyone should have access to the code that drives a piece of software. And not only should you be able to see the code, you should also be able to use, adapt or refine, and distribute the code—or use it to create new tools of your own.

Initially started by a small group of maverick programmers, open source has become a revolution that has changed the way software—and many other things—is made. The pioneers of open source discovered that not only was their

software *free-er*, it was often better too. It turned out that the informal, unpaid community of programmers sharing tips and tools and working for the good of the community was producing software as good as or better than the large corporation's.

Open source has worked so well, in fact, that I just saw a headline on ZDNET that asked if open-source software was beginning to challenge the mother of all technology companies: "Are Microsoft's Problems Due to Opensource?"<sup>1</sup>

The Open Source Initiative's Web site ([www.opensource.org](http://www.opensource.org)) describes the advantages of opening up: "People improve it, people adapt it, people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing. We in the open source community have learned that this rapid evolutionary process produces better software than the traditional closed model, in which only a very few programmers can see the source and everybody else must blindly use an opaque block of bits."<sup>2</sup>

The success of open-source platforms such as Gnu and Linux has inspired some businesses to get on board. Hang on, you say, how does a business that gives away its code stay in business? Here are two possibilities: Companies give away the software but charge for maintenance or support. Or, they hope that the software will encourage people to purchase a related project. For instance, the gaming software itself may be open source, but you still need to purchase the processor and controls to run it. Plus, open source saves businesses money because they no longer need to employ an army of programmers—they get help from the people who use it.

*Open* does not always mean *free*. Sometimes software or products that are developed on an open-source platform are moved behind closed doors. Or part of their code is open source and the rest is proprietary. Google may be the world's largest open-source company—it is built using Linux. Although Google keeps many secrets to itself, it also actively encourages users to adapt and modify the public facing code. In fact, Google sponsors a competition for the biggest innovations in open source: \$25,000, and the competition is open to anyone. Google recognizes that what is good for the open-source community is also good for Google.

The founders of the open-source movement believe that we (meaning them, you, and I) will take over the world. Wikipedia is a great example of an open-source product being used to promote the open-source philosophy. Wikipedia has set out to create an open-source encyclopedia of all human knowledge—anyone can view an entry, edit, or create an entry. Because most facts link to other pages, you can always trace a piece of information to find out more or see who added what piece of information when. And, if you want to make a Wiki of your own, you can go to the "view source" section and copy the code. Wikipedia now has more than 1,500,000 entries in many different languages.

But what does this have to do with design? The design of systems that allow and encourage people to work together has been the defining challenge of the last few years. Although it may not look like it, Wikipedia is a brilliant example of digital design. It is easy to navigate, easy to add

---

<sup>1</sup> Dana Blankenhorn, ZDNet.com, April 28, 2006.

<sup>2</sup> [www.opensource.com](http://www.opensource.com)

to, easy to access, and, most important, actually useful (you might be surprised at how often this last point is overlooked). Other projects that fit in this category include Flickr, MySpace, and, of course, about 50 million blogs and counting. With this kind of competition, it is not always easy to gain what the pros call *mind share*—people need to find your site, remember it, and want to come back.

The success of these projects has inspired other open-source projects in different disciplines. There are a number of open-source projects related to science—these encourage scientists to share data and methods, not just results. There are some movements that advocate an open-source government.

Linus Torvalds, the founder of one the most successful open-source platforms, Linux, says, “The future is *open-source everything*.” With rumors circulating that the Chinese government is switching to an open-source platform for all its computers, he may just be right.

## Talk Like Us

One of the main ideas behind open source is that having more people contributing to a project is better. Of course, one of the largest barriers to getting more people involved in programming has been programming languages, which, like any language, can be difficult to pick up.

That may not be a bad thing. The reason languages are difficult, in addition to some rote memorization, is that they force you to think differently. Languages affect more than just what you can get done or how quickly you can do it. Designer and programmer Jurg Lehni told me, “When working with different programming languages, it is quite interesting to see how the different paradigms affect both my way of thinking and the outcome.”

Still, programming languages are pretty unforgiving—put a comma instead of a semicolon and you may be left with a blank screen. This is beginning to change, says Lehni. “Recently, there have been a lot of exciting things going on around Ruby, a programming language from Japan. Ruby is very flexible, elegant, and sometimes astonishingly close to human language.” A dialect of Ruby called *Ruby On Rails* has been used to create groundbreaking networking applications. Looking ahead, it seems very likely that programming languages—or at least some of them—will continue to evolve to resemble “natural” or human language. As a result, more people will be able to participate in creating and sharing digital tools.

## A Designer by Any Other Name . . .



Define digital designer. Given the growing intersection of graphic design with Web, time-based media, information design, and associated disciplines, including writing and producing (as well as the blurring between fine art and design), who and what we are is becoming more complicated to define and, therefore, to name.



Yet it wasn't always this confounding. Before W. A. Dwiggins famously coined the term *graphic design* in a 1922 Boston newspaper article as a means to describe the wide range of jobs he personally tackled, *commercial artist* was the accepted label for the interrelated acts of drawing and laying out. Dwiggins, however, was a true jack-of-many-graphic-trades, including, but not exclusively, illustrating books; composing pages; designing typefaces; producing calligraphic hand lettering, stencil ornament, book covers, and jackets; creating book interiors and title pages, advertising, and journal formats, along with handbills, stationery, labels, and signs; and writing his own critical essays, fiction stories, and marionette plays.

Dwig (as he was called) wanted to distinguish his prodigious activities from less prolific commercial artists, so he coined a term that was uniquely his own. Graphic design, originally derived from, but much broader than, graphic arts (signifying drawing and printmaking), defined such an esoterically personal pursuit that he could not have predicted that decades later it would become the standard professional nomenclature. Although he never actually called himself a graphic designer, this coinage was cast like bread upon the sea and eventually washed up on professional shores.

When asked if they called themselves commercial artists, not one member of a recent graduating class of design students raised a hand, but surprisingly, only two-thirds of the students embraced the term *graphic designer*. Over a decade ago, when design schools and design firms started affixing highfalutin monikers to academic degrees and business cards, the most common newbie was *communications design*, which, along with *graphic communications* and *visual communications* (or *Viz-Com*) seemed to address the transition from old to new digital media.

In 1975, when computers first began to be an integral part of design—and well before most people thought of setting PCs in the middle of their desks—Richard Saul Wurman gave the field his quixotic appellation *information architects*.<sup>1</sup> In the mid- to late 1990s, when the Web became a dominant presence in design practice, this tag became much more commonly applied. Other terms now include *user interface designers*, *human-centered interface designers*, and *experiential interface designers*.

Currently there is a schism between *digital designers* and a newly coined, curiously pejorative affixation, *conventional designers*, which indicates solely print orientation. The term *conventional design* was originally uttered by a guru in the Web standards movement, who was making a huge distinction between Web designers and print designers, who, by implication, were designosaurs (try that on your business card—or Web site). If graphic design is synonymous with print, and print is conventional, then a priori anything in the nonprint realm is unconventional.

Indeed, the Web and other digital platforms are the proverbial new frontier. With the evolutionary onslaught of new tech already upon us, the day may come when designers will be called ologists, as in designologists, typologists, or interfaceologists.

In a few years it might not be farfetched that the design academies and profession will sweep out all the dysfunctional nomenclature for new terminologies that alter outside and inside perceptions of what we do and who we are. Which, after all, makes sense in this radically integrated new digital media world.

<sup>1</sup> [www.acm.org/ubiquity/interviews/r\\_wurman\\_1.html](http://www.acm.org/ubiquity/interviews/r_wurman_1.html)