The Dark Side of Affordances?

by Brian R. Johnson, Associate Professor Emeritus

The theory of *affordances* is rooted in the work of J. J. Gibson, a psychologist working in the area of visual perception. Design and Human Computer Interaction (HCI) theorist Donald Norman introduced the idea to the digital design community in the late 1980s in his book now known as *The Design of Everyday Things* (cover shown at right). There are quibbles about the definition, but it is now generally understood “to refer to the actionable properties between the world and an actor (a person or animal)”†. In other words, affordances suggest or allow alternative actions that might be taken. The teapot on the cover of his book violates our expectations and experience with pouring hot liquids—it does not afford pouring. A more familiar way to think of this, for architects, is in terms of “ease of use”. The door shown in the next image is ambiguous about where it should be pushed—it is not easy to use.

Of course, what is easy to do depends on the population of users (the elderly may have more trouble twisting off bottle tops than you do, for example, or seeing well at night), and the culture or experience-base (which may be why kids seem better able to set VCR clocks?).

This theory has gained a following in the software design community. Our expectations of the physical world are established through years of childhood hefting, pouring, and pushing. New

† [https://jnd.org/affordances_and_design/](https://jnd.org/affordances_and_design/)
physical objects pretty much have to play by the same rules. However, the virtual worlds of software are entirely constructed of human design decisions. Significant expertise and thought needs to go into making software, teapots, and buildings “easy to use”, and the theory of affordances helps us understand and appreciate the user experience. It also helps us understand how users misuse or make novel use of an object or environment (e.g. using a screwdriver to pry something, since it has a handle and gives leverage).

What we learn when we tackle a new piece of software or a new operating system are the affordances of the new system. Learning the opportunities inherent in a new system (all those commands, tips and tricks) is essential and important—it enables us to establish useful chains of action to get work done. But we need a “big picture” view of the problem too, because in the process of learning, we sometimes learn action plans that aren’t “wise”. For example, at the UW I teach a course in web design. I occasionally see students create web-pages which use HTML image resizing to make very large images into images 1/10th or less their ‘real’ (pixel) size for display in their web page—a strategy that works fine on our (very!) fast campus network, but is painfully slow for viewers outside the UW because the browser must download the whole file before it scales it down and displays it. In this situation the network affordance (speed) masks the need for an alternative approach to the problem if a robust well-designed website is sought. Affordances, like ease of use, are usually seen as positives. Masking likely problems is one way that affordances manifest a darker side.

*Action* in response to a particular situation is also a characteristic of professional practice. When students are learning design, they learn what actions are appropriate to a particular stage of the design process, pick up strategies for resolving apparent conflicts between design goals, and learn what opportunities materials and configurations afford (e.g. different column spacings possible with different structural systems). We could say that they learn the affordances of the design process and tools. We could also say we learn to design in efficient and productive ways.
In the current day and age students expect and are often encouraged to utilize digital design tools in their design tasks. These programs exist to help people who are designing. Just like the SketchUp screen shown here, they’ve got lots of affordances built in. There’s some sort of work area, and a menu of options—things you can do to the data you’re creating. Obviously, you can't do what the program doesn’t allow you to do, or help you do. That’s why you’ve studied the software. But there’s a trap here too. The subtle message in any piece of software is that the things you can do must move the design forward—that the affordances of the software are all you need—why else would it be used all over town? Unfortunately, in a design crisis of conflicting goals, it is unlikely that spinning the model around will reveal the solution, or that zooming in on some other part of the building and adding furniture will make the problem go away. Designers need to find a way of framing the choices available to them in terms of alternative action plans, and then make a choice. The affordances of the software are about active interaction with the 3D model not about interaction with the design, nor contemplation of the results. They seductively offer action when you need design.
Sometimes you need to step away and contemplate a diagram of the choices, or some other representation, rather than the geometry. That’s the big picture that the increasing hegemony of computing and software may mask. Those icons are seductive. The challenge is to learn how best to use these powerful tools to help design, when what they want you to do is interact. Sometimes not clicking is what’s really needed.

*A version of this essay was originally published on the Department of Architecture blog page: January 28, 2013.*