C stands for Creativity

Kostas Terzidis, Ph.D.
School of Arts and Architecture - UCLA

Programming is a way of conceiving and embracing the unknown. At its very best, programming goes beyond developing commercial applications. It becomes a way of exploring and mapping our own way of thinking. It is the means by which one can extend and experiment with rules, principles, and outcomes of traditionally defined architectural processes.

In developing computer programs, the programmer has to question how people think and how mental processes develop and to extend them into real dimensions through the aid of the computers. In other words, computers should be acknowledged not only as machines for imitating and appropriating what is understood, but also as vehicles for exploring and visualizing what is not understood. The entire sequence of specifying computer operations is similar (albeit not equal) to that of the human thinking. When designing software, one is actually transferring processes of human thinking to a machine. The computer becomes a mirror of the human mind, and as such, reflects its thinking.

A few weeks ago I was at a conference that investigated the future of computers in Architecture. I had expected that the panelists would address the opportunities presented to architects and designers alike by the advances in computer aided research. Instead, most everyone seemed interested exploring existing programs, as opposed to holding a philosophical position driven by their own concepts…… At that time I asked a question to a panel of experts about the necessity of designers to know how to program computer code. The answers that I got from them were very surprising to me, ranging from "what does programming have to do with design?" to "yes, design applications should be customizable". At that point I realized that the question should have been "how much programming should the designer know?"

You may already have deduced that I do think that programming is an important part of design education and practice. Programming involves more than simple problem solving, because it is the only way to use the computer to its full capacity, and for challenging known facts. Programming is the vehicle for obtaining new knowledge, for seeing things that cannot be seen, and for taking your fate, as a designer and architect, in your own hands.

Let me give you an example of a personal experience. This example deals with the very basics of architecture: perspective and three-dimensionality. As we all know, any CAAD program will allow the designer/architect to project into space any object/point, and will be able to render it accurately, as long as the designer/architect does not challenge the very basis of the architectural projection: that of a projection being always bound to a formula of positive numbers.

For example, the mathematical formula for a perspective projection is $f(x,y,z)=(x*t, y*t)$ where $t = d + d/z$ and d is the distance of the user to the projection surface. What if I give d a negative value? Can you imagine what would that look like? Can you draw the result on a piece of paper? (it is just a simple formula, isn't it?) Do you know of any CAD application that would allow you to mess around with the perspective projection? I doubt you would find any such application unless somebody gives you the application's code for you to change. But that would involve two things: the designer/architect knowing how to program and the developers giving them the code.

In reality, there is an unraveling relationship between the needs of a designer/architect and the ability of a specific program to address these needs at all times. This can be attributed to a number of factors. First, designers are never really taught how to program (one needs to look no further than the question /answer "What does programming have to do with design?"). Schools do teach students how to use CAD tools, how to play around with applications, but they do not adventure into teaching the language, structure, philosophy, and power of programming.

Secondly, CAD developers rarely release code. You will be asked what you want, you will be offered interfaces for customization but you will not be given access to the code. For good reasons, code is proprietary information, and information is power. So, if a designer/architect wants to mess with the perspective formulas, they will need to write the modeling, interface, display, optimization, and debugging modules on their own. How many people who either have the time or the know-how to do this do you know? When are we going to see a Linux-like CAD system? When are we going to start a community of designers/architects/programmers sharing common code, for the advancement of CAAD?

I tend to believe that now, a designer/architect's creativity is limited by the very programs that are supposed to free their imagination. There is a finite amount of ideas that a brain can imagine or produce by using a CAD application. If a designer/architect doesn't find the tool/icon that they want they just can't translate that idea into form. And whenever they see a new icon (lets say "meta-balls") they think they are now able to do something cool. But are they really doing anything new? If a designer knew the mathematical principles and some of the programming behind the newest effects, they would be empowered to always keep expanding their knowledge and scholarship by always devising solutions untackled by anybody else. By using a conventional program, and always relying on its design possibilities, the designer/architect's work is sooner or later at risk of being grossly imitated by lesser-devised solutions. By cluttering the field with imitations of a particular designer's style, one runs the risk of being associated not with the cutting-edge research, but with a mannerism of architectural style.

In this light, there are many designers claiming to use the computer to design. But are they really creating a new design? Or are they just re-arranging existing information within a domain set by the programmer? If it is the programmer who is asking first all the questions, who is really setting the parameters and the outcome of a good design? We saw already the I-Generation (Internet-Generation). When are we going to see the C-

Generation (Code-Generation) -- the generation of designers/architects that can take its fate into their own hands?. . .