How Should Digital Media be Taught?

by Kostas Terzidis

Note: The following text is based on an interview conducted on 04.14.07 by Junfeng (Jeff) Ding, Senior Designer at Hillier International Architecture, NY and Xiaojun Bu, MArch student at Graduate School of Design (GSD), Harvard University.

Contrary to common belief, computation is not really a goal, but rather the process of arriving at a potential goal. There is a distinction between the visual appearance versus the essence of something. For example, consider the case of a curve. A curve, of course, is a geometrical object that has a visual manifestation, i.e. it looks round, elastic, and soft. But that is just a phenomenon that appeals to our eyes. Behind a curve lies also the mathematical process that defines, describes, and controls it. That has a certain computational complexity that allows the computer to respond to its real time behavior, so that it will look curvy or flexible to your eyes. So the complexity of the curve is something hidden in the computational process. Yet, there is something even deeper than that. That is, the actual complexity that even though it is based on logical arguments, its quantity and articulation is so extreme that it goes beyond one's ability to understand it. Perhaps because humans have a limitation by nature, they just don't understand immediately the complexities involved. Even if many scientists are gathered, still each one of them is limited and so is the whole group. Even if it is split into smaller pieces it still doesn't get understood.

Occasionally, we comfort ourselves thinking that even though some problems are extremely complicated, they are so only in the sense that it would take us a long time to solve. But then again how can that claim be true when we do not understand the problem in the first place; and who is going to negate that claim when we humans are the only judges. So the complexity referred to here is not something remote or abstract but can be found amply in everyday life; for instance, our own bodies are complex structures that we do not know how they work exactly; and yet we are them. Nature is complex, but it can be argued that its processes happen in a certain computational way. The structures of cells, organs, organisms, or even chemical or social phenomena are dominated by processes the mechanics of which we do not understand. There are levels of complexity that need to be understood through some sort of a methodology. One such methodology is computation. In that sense, computation is actually a means to reach a goal, not the goal itself. So it appears possible that computation can be used as a complement to one's own inability to fathom something that is beyond one's understanding, not to do things one already knows.

In my courses, I am trying to sensitize the students about the possibility that there is more than just application driven processes, that is, processes where somebody already provides us with the tools and we just use them. That is to say that, when one uses **form•Z**, Rhino, 3D MAX, or whatever, in reality one is replicating a set of methods that somebody already has done in advanced. In other words, somebody has already assumed that you are going to make a line, and has customized the line command in a way that it is convenient to you. And that convenience I am afraid you pay later on because you're in a way driven to make a decision that you would not have necessarily made, had the same design be done by paper, or more importantly, had it been done by a truly computational process, such as scripting or programming.

The problem is that that decision was not yours. Perhaps an easy analogy is the paradigm of a pool, i.e., one is given the ability to swim, but in a small pool. And then one is able to write one's own scripts and gets more freedom, perhaps now swimming in a lake. And later on one can go on to the ocean. And then one is faced with infinite freedom. Because there is no constraint on the size of the pool that one has been placed in, when one thought that he or she was free, but really was not. I often use the phrase "form follows software", in the sense that software affects the way one thinks. Unfortunately (or fortunately), different software implicitly enforces one to make stylistic decisions. In that sense, it is easy to distinguish a design made in **form•Z**, because it is possible to discern certain characteristics that are stylistically provided by **form•Z**, which means one is actually abiding, almost as a mannerism, to that particular software that actually in a way manipulates the way one thinks, decides, and designs.

In my GSD classes, we are trying to have the students think in the reverse way. It is a fairly complicated process, because for them it is completely unexpected. Yet it is extremely useful because it becomes the first time that they get acquainted with the computer not in a friendly customizable spoon-fed fashion, as in "do this for me." But it is more along the lines of "I need to first find out the logical and mathematical principles, the computational elements, and the relationships to articulate them into things that could be architectural." As in swimming in the ocean, at the end they cultivate their ability to do the things that they really want. Of course, it is hard and it needs lots of work, but I think the results are truly exceptional because they can design not by nursed copying or imitating but by creating; in the true sense of the word.

Any criticism of the current state of how computers are being used or taught is perhaps premature since my approach is too early to conclude. However, these classes at the GSD are an indirect criticism or comparison to other schools and practices, because as mentioned earlier, when one uses ready-made software, such as in modeling applications, they do things easily and fast, and so one tends to be seduced, drawn into, and follow because it is easy, fascinating, and produces results fast and impressively. So faculty and students like it, because it gets things done faster and more efficiently. Yet, at the end, while they may think that they are designing computationally, in reality they are not. They are not really using their minds, logically speaking. They are not challenging the discrete mathematical entities by manipulating them through logical operations which is what computation is. They are just moving the mouse on the screen, by following a preset process being given to them by the programmers that sell these applications.

It is actually an economic rather than an intellectual relationship. For the price of software there is an

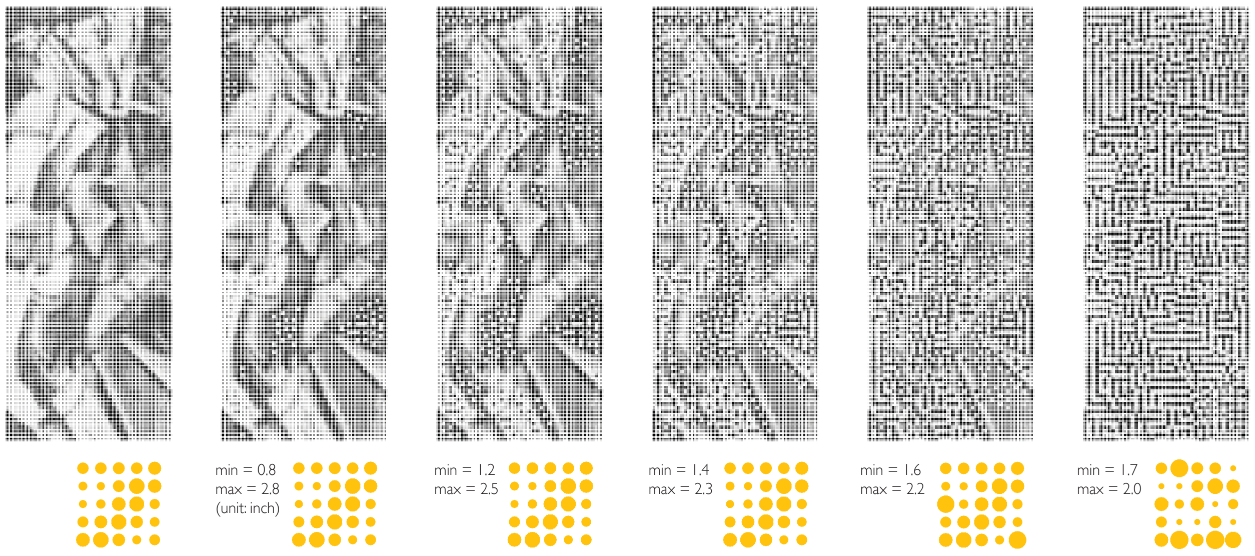

Figure 1: Cellular automata studies by Zhou Xu, MArch II. An array of circles whose radii are based upon the RGB values of pixels from Marcel Duchamp's painting: "nude descending a staircase." Then each circle is judged twice by its eight neighbors' average radius value; if it is less than a designated minimal value, the area can be regarded too light, hence the target circle is replaced by a larger one, reversely, if the area appears to be too dark, then a smaller circle would be the replacement.

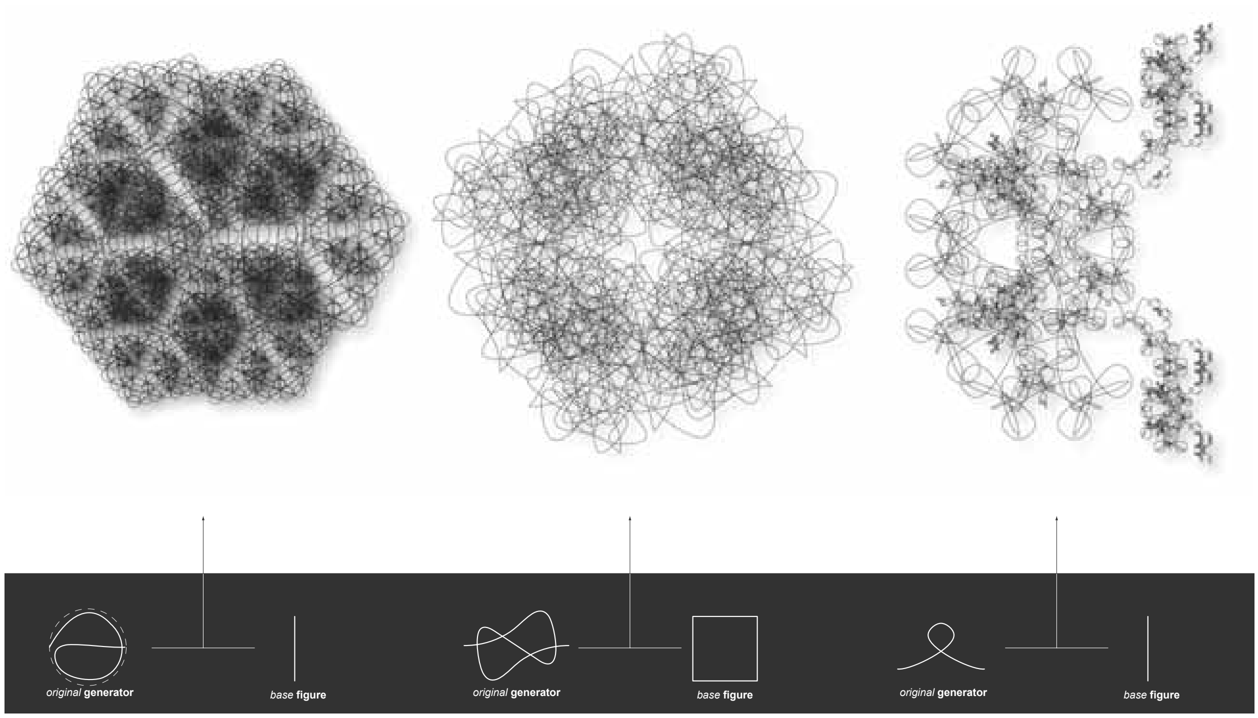

Figure 2: Fractal studies by Kei Takeuchi, MArch I. Each curve segment of a base shape is replaced by a curve called generator. The results are shown above for multiple replacements.

investment return. But in reality they are paying the price of "eye candy" that they get through this kind of process, which I refer to it as computerization. In other words, this relationship reveals that computation is not about the value of a computer and its software, but rather about a mind that thinks using arithmetic and logic as if it is a computer. Of course, when one uses **form•Z**, there is surely a computational process somewhere, but it is not there because of the computer itself. A computational process does not need a computer necessarily. Of course, computers, as data processing machines will help, but it isn't solely the computer itself. It is a logical and arithmetic device, which has nothing to do with the computer the way it is comprehended today. It is not a little gray box with a pixel screen. Rather, it is a flow of immaterial information. There is a distinction. And the problem is that a lot of people don't know it. A lot of people think of the computer itself, as if somewhere inside the computer, something magical is happening. Instead, it is in one's mind. You are the one who makes it. So, in a way, we should be doing design the right way, and not be affected out by software companies. Although I have nothing against them--after all they are doing their job--, and so should we. We can still make parti-design, we can still make diagrams with truly computational methods using numbers and relationships, and then use the computerized techniques for rendering, presentations, etc., which I think they are very good for those purposes. But I don't believe that the pixel-perceived image that one makes moving the mouse is also computational or algorithmic; because it is not. There is no computational process in the way it was designed. That is my distinction.

Kostas Terzidis is an Associate Professor at the Harvard Graduate School of Design. His current GSD courses are Kinetic Architecture, Algorithmic Architecture, Digital Media, Advanced Studies in Architectural Computing, and Design Research Methods. He holds a PhD in Architecture from the University of Michigan (1994), a Masters of Architecture from Ohio State University (1989) and a Diploma of Engineering from the Aristotelion University in Greece (1986). He is a registered architect in Europe where he has designed and built several commercial and residential buildings. His most recent work is in the development of theories and techniques for algorithmic architecture. His book Expressive Form: A Conceptual Approach to Computational Design published by London-based Spon Press (2003) offers a unique perspective on the use of computation as it relates to aesthetics, specifically in architecture and design. His latest book Algorithmic Architecture, (Architectural Press/Elsevier, 2006), provides an ontological investigation into the terms, concepts, and processes of algorithmic architecture and provides a theoretical framework for design implementations.