Harvard School of Design
Cambridge, Massachusetts

Tool-Makers vs Tool-Users (or both)?

by Kostas Terzidis

There is a fundamental difference between algorithmic and CAD-related (or inspired) design. The difference is not only technical, representational, or graphical but also scientific, rational, methodological, and as such, intellectual. Algorithmic design employs an abstract symbolic language for representing ideas, concepts, and processes to be manipulated by a computer. It is a way of thinking, whose power is derived not only by the articulation of thoughts within the human mind but also, most importantly, by the extension of those thoughts using computational devices. In contrast, CAD related design is a graphical manipulation of predetermined elements or processes given to the designer as tools but whose potential capabilities have already been set in advanced. Every time a CAD programmer creates a new tool to be added to the palette, the programmer predetermines what the designer may need. This process involves at least two paradoxes: first, the intellectual effort to conceive, picture, and determine the use of a tool, involves by definition decisions, opinions, and predispositions that set limits to its use by others. The imagination of a particular person has its unique limitations confined however only for that person, not necessarily for others. Second, while the programmer is able to provide those tools that are believed to be needed, at the same time the programmer is unable to provide the means to create the tools that are not believed by the programmers to be needed (but may be) by the designer.

Algorithmic design involves symbolic languages and as such provides the means to create anything whether needed now, or not yet. CAD is about the pre-established needs of a designer who should act in a certain conventional way of thinking. William Mitchell argued that "architects tend to draw what they can build, and build what they can draw." Using CAD tools involves by necessity a Whorfian effect in which the designer is bounded to the potentiality of the new tools, who, in turn, bound the mind of the designer to think in a certain way in order to take full advantage of those tools. This unfortunate circular situation can only be eliminated when the designer and the CAD programmer are one and the same person. Then, the mind that designs is also the one that invents the tools that allows the mind to exceed its own thoughts. A tool is not only an instrument that is used in the performance of an operation whose purpose is known, but also a vehicle

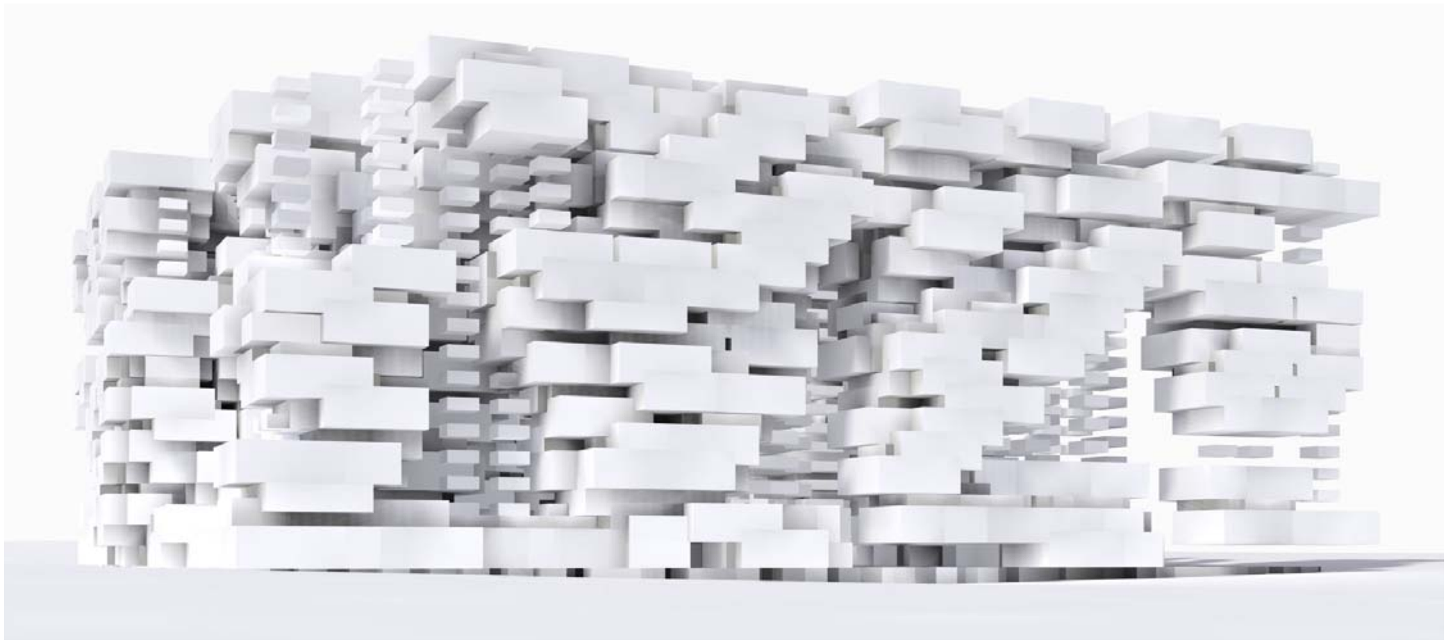

Figure 1. **A housing arrangement for 200 units is constructed using a scripted algorithm based on cellular automata theory. While the final form of the arrangement was unpredictable, the programmatic requirements were predicted to be satisfied**

to help conceive operations or ideas that are not known in advance. The difference between a conventional and a computational tool is in the intellectual nature of the latter. Computation as a process is similar to the way the human mind works, and while it may not have intention or purpose it does perform in a similar manner.

The dominant mode for using computers in design today is a combination of manually driven design decisions and formally responsive computer applications. The problem with this combination is that neither the designer is aware of the possibilities that computational schemes can produce nor the software packages are able to predict the moves, idiosyncrasies, or personality of every designer. Therefore, the result is a distancing between the potential design explorations and the capacity built into computational tools. Designers often miss the opportunity opened up to them through digital tools, merely because of lack of understanding that computation can be part of the design process as well. While some digital designers are claiming to be great fans, users, or explorers of digital design, a lack of knowledge on what really constitutes digital design contributes toward a general misunderstanding; the use of computer applications is not per se an act of digital design.

Digital, in the true sense of the meaning, is about the reduction of a process into discrete patterns and the articulation of these patterns into new entities to be used by a computer. Digital is an achievement of the collective organizational properties of computers not the intrinsic nature of the appearance of their products. In other words, digital is a process not a product. If it is seen as a process, then the emphasis is placed on understanding, distinguishing, and discerning the means by which design can enter the world of computation, and not the other way around. The world of computational design is quite different from the manual world of design. Terms, concepts, and processes that are seen as inconceivable, unpredictable, or simply impossible by a human designer can be explored, implemented, and developed into entirely new design strategies within the digital world. Instead, what is happening is the use of computers as marketing tools of strange forms whose origin, process, or rationale of generation is entirely unknown and so they are judged on the basis of their appearance often utilizing mystic, cryptic, or obfuscating texts for explanation.

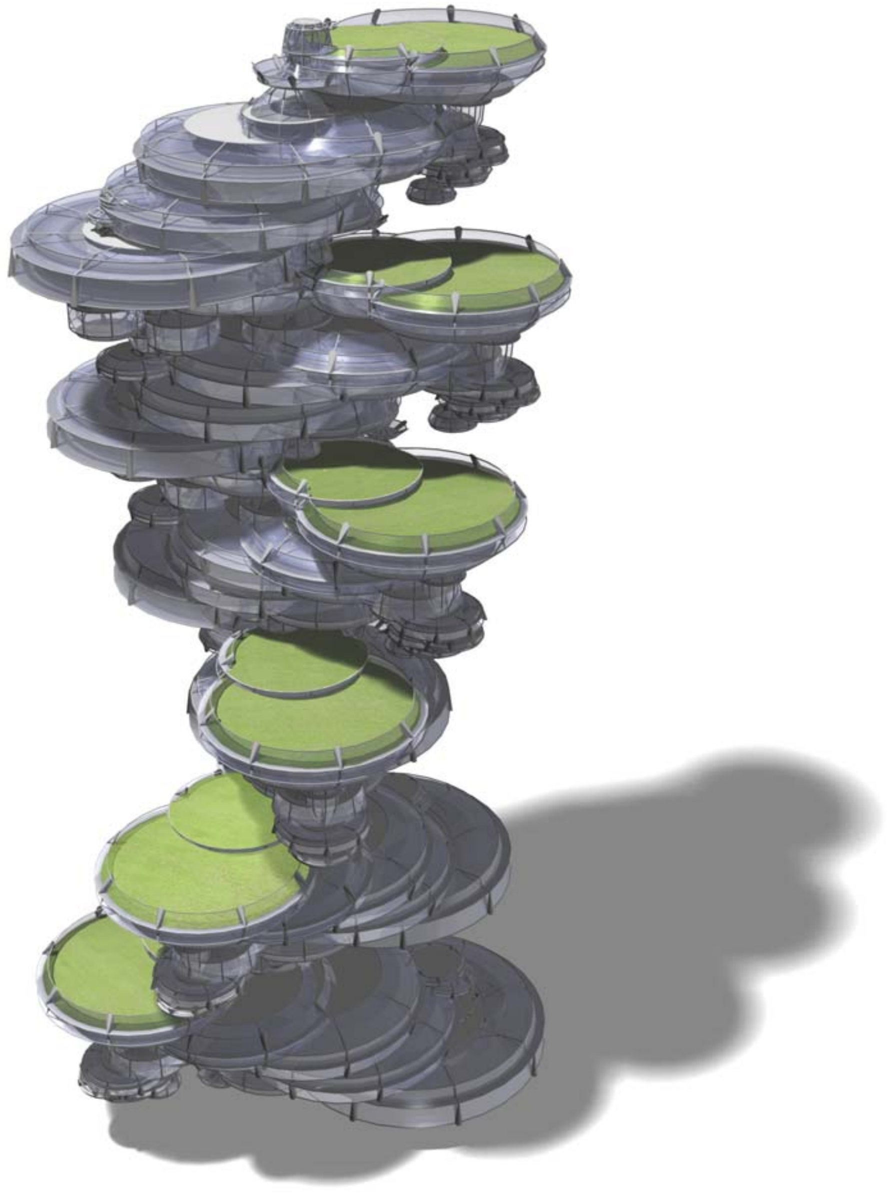
Figure 2. **A building configuration constructed gradually using a scripted algorithm.**

The problem with algorithmic logic in design is that fixed interrelationships between numbers and concepts appear to some designers as too deterministic. In fact, many designers are not interested in the mathematics of a design composition but rather in the composition itself. While this position may be interpreted as a defense mechanism against the possible rationalization of design, yet it becomes also an obstacle in exploring the limits of a possible rationalization of design.

Computer systems that are referred to as CAD systems are in essence collections of algorithms each of which addresses a specific graphical design issue. A user of a CAD system, i.e. a designer, makes use of these algorithms without knowledge of how they work and consequently is unable to determine the full value of their potential. While CAD systems helped designers significantly to organize, speed up, or communicate ideas using high-level commands, only a few CAD systems offer the means to combine those commands algorithmically (i.e. scripting, API, or open-source) in ways that would allow one to explore “out of the box” possibilities or to break

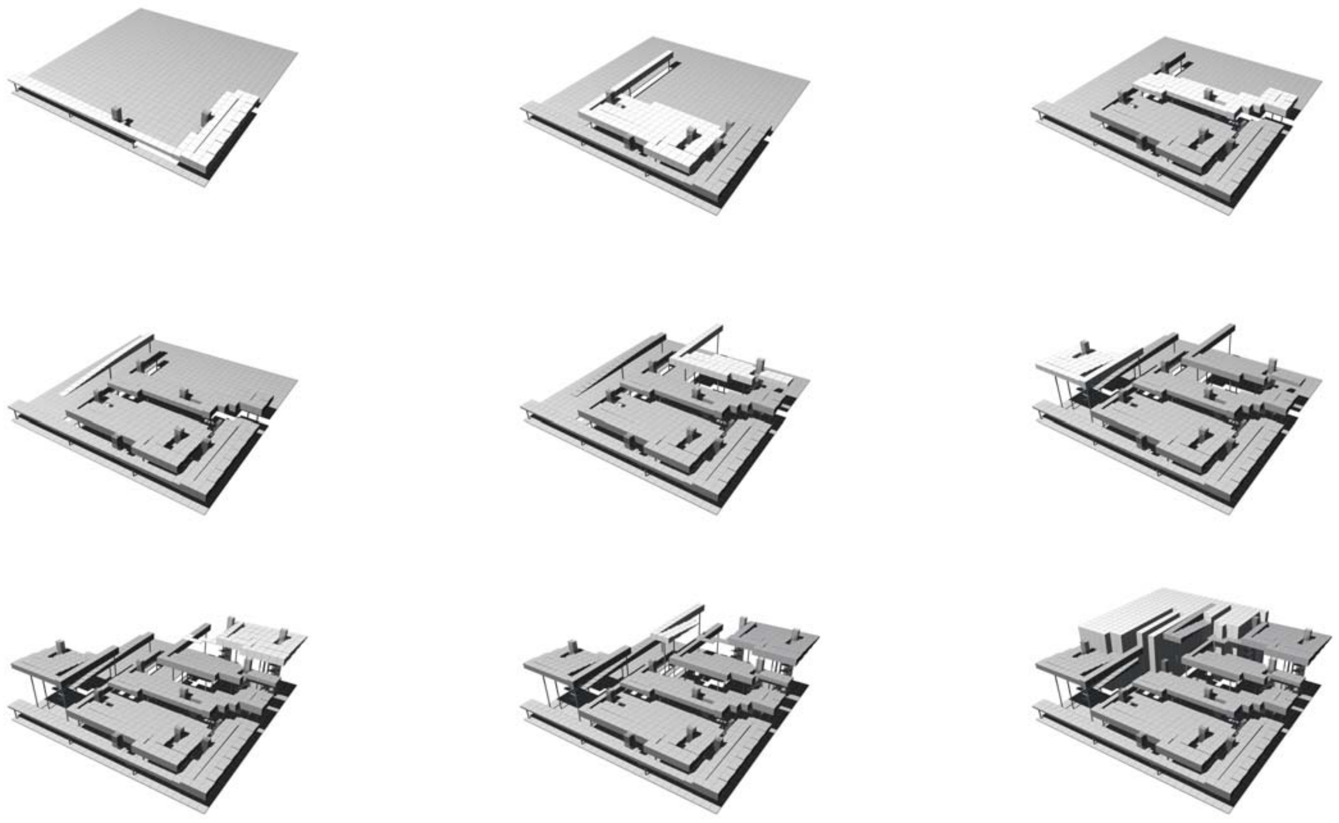

Figure 3. **A library constructed using a scripted algorithm based on stochastic search. The phases of construction reveal a progressive evolution of form based on iteratively satisfying the program's spatial and relational constraints.**

down the commands in ways that would allow one to explore what is "under the hood". Further, very few designers have the knowledge to understand the computational mechanisms involved in a CAD system, or, reversely, very few CAD developers are also equally accomplished designers.

Both non-users and users agree that the effect computers will have on design whether desirable or not will be significant, profound, and far-reaching. This agreement is based on an important yet peculiar relationship between design and its tools. It is apparent that design is strongly depended on the tools utilized and, reversely, tools have a profound effect in design. Traditionally, this dependency is controlled by the human designer who decides which tool is to be used when and where as well as the range of possibilities a tool has for addressing, resolving, or accomplishing a design task. Further, it is possible that the use of tools may also have further implications in the process of addressing a task: just because a tool is available, a task is now possible, or, further, a tool implies a task. However, a problem arises when the tool is not entirely under control of its user. In the case of a computer as a tool, the results may be unexpected, surprising, or unpredictable even by the users themselves. While such moments of surprise may be advantageous, enlightening, or perhaps even undesirable, they do exhibit a theoretical interest because they challenge the basic premise of what a tool is or how a tool should behave. Further, such behavior may lead to alternative ways of executing the task that were not intended and may lead to results often superior than intended.

Traditionally, the dominant paradigm for discussing and producing design has been that of human intuition and ingenuity. For the first time perhaps, a paradigm shift is being formulated that outweighs previous ones. Algorithmic design employs methods and devices that have practically no precedent. If design is to embark into the alien world of algorithmic form, its design methods should also incorporate computational processes. If there is a form beyond comprehension it will lie within the algorithmic domain. While human intuition and ingenuity may be the starting point, the computational and combinatorial capabilities of computers must also be integrated.

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 The 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.