Digital Design: Ideological Gap or Paradigm Shift?

Kostas Terzidis / Graduate School of Design, Harvard University, USA / Kostas@gsd.harvard.edu  http://www.gsd.harvard.edu/faculty

Abstract. In this paper a series of arguments about the gap between manual and digital practices will be raised, discussed, and addressed in the context of design education and practice to illustrate a discrepancy and the possibility of a paradigm shift.

Introduction In response to the theoretical challenge posed by the theme of this conference a series of philosophical arguments will be discussed that address not only the theme but question the general framework upon which ideas are being formed on the basis of visual appearance through the use of digital media. This framework is not only technical or methodological but most importantly an ideological one.

Within the emergence of digital processes, it can be argued that certain qualities of the human mind such as those that contribute to what is considered “smart”, i.e. sharpness, quick thought, or brightness, may not be applicable or even desirable when dealing with the computer’s reasoning. What is considered to be smart in the one world may be considered dumb in the other world. Traditionally, the dominant mode for discussing creativity in architecture has always been that of intelligence, talent, and intuition, where stylistic ideas are pervaded by an individual, a “starchitect” or a group of talented partners within the practice.

In contrast, computation is a process, the result of which is not necessarily credited to its creator. Algorithms, for example, are understood as abstract and universal mathematical operations that can be applied to almost any kind or any quantity of elements. It is not about the person who invented it but rather about its efficiency, speed, and generality. It can be argued therefore that human decision making (i.e. manual design) is based on human decision making and, as such, can be arbitrary, authoritative, and, often, naive in comparison to computational schemes where consistency, generality, and complexity are celebrated. Objectivity liberates design from subjective interpretations and leads towards the emergence of designs that while satisfying the program’s requirements may surprise even their own creators.

These two distinct practices have deep and profound differences that are not only ideological or methodological but most importantly generational. As the two generations are distinguished by a separation in digital education, the older encountering the digital after the manual and the younger intrinsically emerged in the digital from an early age, a problem arises on the understanding, comprehension, and appreciation of design as well as its conceptualization, execution, and criticism.

Generational gap Under the older generation, 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 whatever constitutes creativity and ingenuity is, by definition, human.
While the original goal of CAD was to free the designer from repetitive, tedious, or time-consuming tasks, it also sought to empower the designer with the means to explore beyond the traditional framework of manual design. The desire to implicate a digital mode of thought within the design process was intimately linked to the nature of computation and its close association with that of design. If design is to be considered a systematic, finite, and rational activity then a computational scheme could be devised that would encapsulate, codify, and reflect the process. Further, such a scheme could be transferred and processed using a computational device such as a computer. The initial thought was that because computation employs complex processes, such as simulation, optimization, permutation, or transformation, that such processes could be applicable, useful, if not catalytic in addressing design problems. However, due to the complex nature of the processes, very few designers were in a position to understand and implement them in a meaningful way in design.

As most of the researchers in CAD were primarily concerned with the technicalities of converting design ideas into digital tools, none, if any, was also concerned with using those tools to actually design. Apparently, the design sensitivities involved in creating a tool are not the same as in those involved with using one. The unprecedented potentiality of the new CAD tools brought a high expectation in how to change the way designers work, create, and think. Therefore, a paradigm shift was sought from within the designers’ world, one that would occur by employing the newly created CAD tools. However, it may be argued here that the long awaited paradigm shift occurred not in the designer’s mind but in the programmer’s mind. It is the programmer that invented the tool and set out the workspace, capabilities, and limitations for the designer to work within. CAD software developers are meta-designers, i.e. designers of design-systems. In contrast, the traditional designers-turned-digital are merely spectators to a world that extends beyond their comprehension. For the last two decades, beginning with Eisenman’s visions and Lynn’s curvilinearity and continuing through an overwhelming plethora of so-called digital design studies, architects have been primarily concerned with the formal manifestation of scientific theories using the computer as a medium of expression. Instead of using computational theories as the structural foundation for architectural experimentation, instead they employed humanistic philosophical theories of the 60’s and 70’s to explain the complexity of the forms they produced using computers. These practices have attempted to seek for a theoretical foundation of digital phenomena within the scope of classic humanistic methods, i.e. observation, explanation, or interpretation. While such methods are among the fundamental sources of knowledge, they cannot explain the realm of computational phenomena because those extend beyond the sphere of human understanding. Concepts such as randomness, infinity, limit, infinitesimal, or even more elaborate concepts such as complexity, emergence, or recursion are incomprehensible by the human mind not because they are metaphysical, magical, or mysterious but rather because they depend on intellectual means that are external and foreign to the human mind. Instinctively, at the absence of anything else, humans throughout their history have always used as reference tried to overcome their material nature by seeking for concepts and ideas that are out or independent of their own existence. Perhaps for the first time, through the invention of the computer, a device originally intended to serve people, ironically were faced with phenomena that demarcated the limits of the human mind and cast some light into the borders of an alien world.

In search for a theoretical foundation While humanistic approaches praise and celebrate the uniqueness and complexity of the human mind they also become resistant to theories that point out the potential limitations of the human mind. Late modernist, phenomenological, or cultural critical theories differ significantly from those of mathematics, linguistics, or computation in that the former use as reference the human consciousness whereas the latter seek to separate the subject from the object, seeking instead for principles that lay out or independent of human existence. The use of human presence as a witness of phenomena is a strong underlying framework upon which humanistic theories are based. In contrast, scientific theories tend to quantify events objectifying their effect in order to avoid human interpretation. In the architectural theories of the past twenty years, a certain predominant group of theo-
reticians, following the traces of humanistic philosophies, have been seeking to find origins, sources, or connections between humanism and digital phenomena. This approach is understandable and expected especially by a group of people that have a deep understanding of humanistic philosophies. The problem however with this approach is that it doesn’t take under consideration alternative theories, concepts, or methods that are perhaps alien, foreign, and even antithetical to the dominant traditional humanistic philosophies.

Most computational theories become incomprehensible, unintelligible, or incomplete if they are not understood as part of a complementary interaction between the mind and the computer. Because of the external nature of computation, mere reading, studying, or speculating on its theoretical implications is not sufficient enough to grasp its hidden mechanisms. In contrast, actual implementation (i.e. programming) reveals mechanisms, events, or phenomena that defy human explanation. In architecture, this dichotomy was expressed by two antithetical thought camps, formulated by two dialectically opposed ideologies: that of tool-makers and that of tool-users. The first ideology, rooted in the principles of computation, strove to offer the means for design explorations using computers as vehicles. The main protagonists of this ideological camp are software developers, computer scientists, and mathematicians. In contrast, the second ideology sought to connect humanistic philosophies with digital phenomena. In doing so it had to search for ideas or principles within the humanities that may explain or address digital phenomena. For instance, Lynn argues that the plasticity of computer generated forms may be associated with Deleuze’s descriptions of smoothness and continuity, as if software is associated with softness (Lynn, 1999). While this may hold some value at a phenomenal level, it certainly holds no truth at a mathematical level. Polynomial-based curves or surfaces, i.e. nurbs, exhibit a continuous and smooth behavior only when implemented on a computer system. It is the numerical representation, processing power, and display resolution of a computer system that makes the plasticity possible, something unknown and perhaps irrelevant to Deleuze (In fact, Pierre Bézier (1910-1999), the French engineer and creator of the Bézier curves and surfaces that are now the basis of most computer-aided design systems makes no reference to Gilles Deleuze (1925-1995) and vice-versa, even though both lived in the same place (Paris) at the same time). However, concepts such as numerical, processing, or resolution are not human and therefore cannot be credited. Instead, at the absence of a sentient identifiable human creator, a philosopher’s position seems more appropriate. In such a way, humanism is credited, praised, and celebrated by its fellow supporters in a self-referential manner. This anthropocentric attitude is even clearer in Lynn’s comparison of a computer to a pet (Lynn, 1999).

The use of the words domesticate and wilderness are characteristic of an anthropocentric and human-dominating attitude rather than a synergistic and collaborative one.

**The Logic of Computation**

What makes computational logic so problematic for architects is that they have maintained an ethos of artistic sensibility and intuitive playfulness in their practice. In contrast, because of its mechanistic nature, a computational process, e.g. an algorithm, is perceived as a non-human creation and therefore is considered distant and remote. Traditionally, the dominant mode for discussing creativity in architecture has always been that of intuition and talent, where stylistic ideas are carried out by an individual, a star, or a group of talented partners within the practice. In contrast, an algorithm is a procedure, the result of which is not necessarily credited to its creator. Algorithms are understood as abstract and universal mathematical operations that can be applied to almost any kind or any quantity of elements. For instance, an algorithm in computational geometry is not about the person who invented it but rather about its efficiency, speed, and generality. Consequently, the use of algorithms to address formal problems is regarded suspiciously by some as an attempt to overlook human sensitivity and creativity and give credit instead to an anonymous, mechanistic, and automated procedure (in response to this discrepancy, the ancient Greeks devised a fair method of acknowledgement of authorship.

The Pythagorean Theorem, the spiral of Archimedes, or the Euclidean geometry is an attempt to give proper credit to the authors regardless of the status of their subjects as inventions or discoveries). In any case,
algorithms are encapsulations of processes or systems of processes that allow one to leap and adventure into the world of the unknown whether natural or artificial. They are not the end product, but rather a vehicle for exploration. What distinguishes these processes from common problem solving is that their behavior is often non-predictable and that frequently they produce patterns of thought and results that amaze even their own creators.

Computation is a term that differs from, but is often confused with, computerization. While computation is the procedure of calculating, i.e., determining something by mathematical or logical methods, computerization is the act of entering, processing, or storing information in a computer or a computer system. Computerization is about automation, mechanization, digitization, and conversion. Generally, it involves the digitization of entities or processes that are preconceived, predetermined, and well defined. In contrast, computation is about the exploration of indeterminate, vague, unclear, and often ill-defined processes; because of its exploratory nature, computation aims at emulating or extending the human intellect. It is about rationalization, reasoning, logic, algorithm, deduction, induction, extrapolation, exploration and estimation. In its manifold implications, it involves problem solving, mental structures, cognition, simulation, and rule-based intelligence, to name a few.

The dominant mode of utilizing computers in architecture today is that of computerization; entities or processes that are already conceptualized in the designer’s mind are entered, manipulated, or stored on a computer system. In contrast, computation or computing, as a computer-based design tool, is generally limited. The problem with this situation is that designers do not take advantage of the computational power of the computer. Instead some adventure into manipulations or criticisms of computer models as if they were products of computation. While research and development of software involves extensive computational techniques, mouse-based manipulations of 3D computer models are not necessarily acts of computation. For instance, it appears, from the current discourse, that mouse-based manipulations of control points on NURBS-based surfaces are considered by some theorists to be acts of computing (Cuff, 2001).

While the mathematical concept and software implementation of NURBS as surfaces is a product of applied numerical computation, the rearrangement of their control points through commercial software is simply an affine transformation, i.e. a translation.

**Paradigm Shift** When comparing contemporary practicing architects such as Thom Mayne, Frank Gehry, and Peter Eisenman it is necessary to overlook many significant and distinguishing differences in order to identify at least one common theme: the use of the computer as an exploratory formal tool and the increasing dependency of their work on computational methods. The most paradigmatic examples of the last ten years invest in computationally generated parts and diagrams. Through computation, architecture transcends itself beyond the common and predictable. In contrast, computerization provokes Whorfian effects: through the use of commercial applications and the dependency on their design possibilities, the designer’s work is at risk of being dictated by the language-tools they use. By unknowingly converting to the constraints of a particular computer application’s style, one runs the risk of being associated not with the cutting-edge research, but with a mannerism of hi-tech style.

A paradigm shift is defined as a gradual change in the collective way of thinking. It is the change of basic assumptions, values, goals, beliefs, expectations, theories, and knowledge. It is about transformation, transcendence, advancement, evolution, and transition. While paradigm shift is closely related to scientific advancements, its true effect is in the collective realization that a new theory or model requires understanding traditional concepts in new ways, rejects old assumptions, and replaces them with new. For T.S. Kuhn, scientific revolutions occur during those periods where at least two paradigms co-exist, one traditional and at least one new (Kuhn, 1996). The paradigms are incommensurable, as are the concepts used to understand and explain basic facts and beliefs. The two live in different worlds. The movement from the old to a new paradigm is called a paradigm shift.

Traditionally, the dominant paradigm for discussing and producing architecture has been that of human
intuition and ingenuity. For the first time perhaps, a para-
digm shift is being formulated that outweighs previous
ones (Eiseman, 1992). Algorithmic design employs meth-
ods and devices that have no precedent. If architecture
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 computa-
tional and combinatorial capabilities of computers must also
be integrated.

References
Cuff D., 2001 Digital Pedagogy: An Essay: One educator’s thoughts on design software’s profound effects on design thinking and teaching, Architectural Record, September 2001. (In this article, Cuff considers that computing is “one of the most important transformations of the contemporary profession” and that today “computing has become a populist skill”). / Eisenman P. 1992, Visions Unfold-
ing: Architecture in the Age of Electronic Media, Ars Electron-
ica 1992. (Peter Eisenman referred to the idea of an electronic paradigm shift in architecture in 1992. He wrote: “During the fifty years since the Second World War, a paradigm shift has taken place that should have profoundly affected architecture: this was the shift from the mechanical paradigm to the electronic one.”) / Kuhn, Thomas S. 1962 The Structure of Scientific Revolutions Uni-

Keywords: Computation, design, para-
digm shift.