Deconstructing the Software Interface: A Critical Close Reading of AutoCAD

Mahesh Senagala

IJAC 2-3 (1 proof) 20/10/04 12:27 pm Page 300

Deconstructing the Software Interface: A Critical Close Reading of AutoCAD

Mahesh Senagala

AutoCAD maintains a nearly 70% market share in the PC-based AEC sector and wields enormous influence over design and production processes in architectural firms and schools. Such an impact is, perhaps, more than what a single building can hope to achieve. The design implications of such a market monopoly are many. Based on Derridean operations of *Deconstruction*, the paper will deconstruct AutoCAD's latent agenda. The paper will do a critical close reading of AutoCAD for its design preferences, spatial conceptions, worldviews, resistances, stratifications and organizational predispositions with respect to architectural design process. For purposes of brevity, this paper will focus on the architecture of AutoCAD's interface. The results of the paper would be the *beginning* of a critical theory that can be employed in the process of software design for design professions.

1. Introduction

Imagine 3,469,400 users diligently staring into the black screen of AutoCAD as it starts up [12]. This is the number of installed bases. The number of users might well be in excess of four million. Three-and-a-half million is approximately the present population of Los Angeles metropolitan area! And that's the worldwide market share of AutoCAD. AutoCAD maintains a nearly 70% market share in the PC-based AEC sector and wields enormous influence over design and production processes in architectural firms and schools. A survey conducted in 2000 by *Geopraxis* revealed that 69% of the surveyed architectural firms use AutoCAD for 2D drawing. Also 42% use AutoCAD for 3D modeling. However, only 13% of AutoCAD's users said their firm's use of AutoCAD for 3D modeling was "Most Effective" [11]. Such an impact is, perhaps, more than what a single building can hope to achieve. The design implications of such a market monopoly are many. The paper intends to fill the void created by the paucity of critical work on this topic which affects millions of architects, students and building users. While texts, buildings and even films have been deconstructed widely, software systems have not been subjected to the same degree of systematic critical "*solicitation*" despite their deeper impact on the society. Also, while the field of architectural design has been greatly supported by a wide body of theoretical knowledge and critical studies, sadly enough there are no similar discourses about software design available to us. Based on Derridean operations of Deconstruction, the paper will deconstruct AutoCAD to reveal its surprising latent agenda. The paper will be a critical close reading of AutoCAD for its design preferences, spatial conceptions, worldviews, resistances, stratifications and organizational predispositions with respect to architectural design.

For purposes of focus, this paper will focus on the architecture of AutoCAD's interface. *The results of the paper would be the beginning of a critical theory that can be employed in the process of software design for design professions.* The paper will question a number of otherwise taken-for-granted performative/prosaic approaches to and notions about the design of computer-aided design software tools.

1.1. Derridean operations: a brief outline

French philosopher Jacques Derrida's work, which spans more than four decades, understandably, cannot be summed up in a couple of paragraphs. I will however, make an attempt to give a very brief outline of Deconstruction if only as a refresher.

Most of the seminal foundational work by Derrida was complete by the early seventies that established Deconstruction as a highly controversial philosophy of philosophies to reckon with in philosophical as well as literary circles. To the English readers Derrida was introduced through two major translations of *Of Grammatology* (1976) and *Writing and Difference* (1978). However, back in 1966, Derrida presented an important paper 'Structure, Sign, and Play' – at Johns Hopkins in Baltimore, a decade before the first translation of *Of Grammatology* appeared. Deconstruction differs from other forms and streams of philosophy through its activist practice. What we find fascinating in Derrida's works is that they are critical acts rather than analytical writings about specific works. Deconstruction's focus is very tight on what it chooses for a critical close reading. Deconstruction, as Michael Benedikt pointed out, "as a rule, proceeds from an essentially conservative definition of what the work at hand is, and limits its actions to what is internal to the work" [1] In that sense, it is understood and assumed (rightly so) that the external forces, contextual parameters and other issues of the larger world find manifestation in the work at hand. Deconstruction focuses on "how a particular text gains meaning" as opposed to "what is meant by that text." It is akin to surgery: focused, rigorous and clinical. However, Derrida's works are seldom clear or easy to read. This is a narrative strategy that Derrida employs to make a point. Derrida proposes no overarching theory, ideology or narrative unlike other major philosophers. In practicing Deconstruction, Derrida often treads a fine line between philosophy and literature.

Derrida is known for his inimitable style, critical attitude and politically subversive mode of writing as a means to expose the binary oppositions and reveal the fictitious boundaries of the philosophical texts. Derrida's operations are not critiques; a critique requires one to stand outside of the subject. Instead, Derrida reads those texts from inside (just as a virus invades the host) and exposes the carefully constructed binary oppositions such as central/marginal, good/bad, being/non-being, presence/absence, God/man, writing/speech etc. He shows how the first term is privileged and promoted by suppressing and playing against the second term. In architecture, the binary oppositions usually manifest as 3D/2D, drawing/writing, center/periphery, scale/scaleless, presence/absence, inside/outside, up/down, space/form, and so on. Due to its inherently subversive nature and the difficulty of working from within a chosen subject, and due to the less than sterling reputation that it gained from certain architectural applications, many of us have come to either doubt or abandon this extremely unique method of critical close reading.

What we do understand from Derrida is that philosophical texts employ the following procedure in constructing their textual edifices:

1. Create strong polarities and distance one from the other (speech/writing)
2. Privilege the first one (speech is good)
3. Quell the second one (writing is evil)
4. Ignore the in-between and the undecidable issues.

One of the major observations of Derrida is that these polar opposites cannot exist in isolation. There can be no "good" without the "evil!" In fact the "good" can exist only as long as the "evil" is maintained, maligned and opposed. Between the black and the white, all the undecidable grays are suppressed and eliminated from the picture, so to speak. Derrida's writings, true to his argument, are dense webs of quotations, references, textual juxtapositions and yet precise productions. For instance, one of the most unreadable of Derrida's works, *Glas*, was made so that its physical volume amounted to 100 cubic inches (10" $\times$ 10" $\times$ 1"). If we realize that in France, the publishing industry follows metric system of measurement, the reader can imagine the precise method behind Derrida's apparent madness.

1.2. Why deconstruction? Why now?

Deconstruction has become much maligned in architectural circles. It is frowned upon in architectural practice and academia alike with a few exceptions. However, often this frowning is done with respect to the stylistic or ideological practices of "deconstructivism" as manifested in the works of such architects as Peter Eisenman, Daniel Libeskind et al. Needless to say, not many academicians, let alone practicing architects, I would surmise, are actually familiar with Derrida's work. Why use the much maligned deconstructivist reading now? Why use it after the notoriety that it had gained during the eighties in architectural circles? Why use an "out-of-fashion" device of criticism now? Why use Deconstruction as opposed to phenomenology or commonsense or structuralism or some other formal or informal approach?

Michael Benedikt had an answer ten years ago to these questions when he used the Derridean reading to critique Louis Kahn's Kimbell Art Museum in Fort Worth, Texas (Benedikt, 1991). He was one of the few architects to have actually employed the Derridean operations to reveal the extraordinary and latent schemata of the Kimbell. He wrote: "Derrida's Deconstruction can mean more to architects (and artists) than a transitory aesthetic or a style, and it should not be allowed to devolve into the esoteric, promotional patter and stylish nihilism that it threatens to do." Further, he pointed out that "Deconstruction's destiny, I believe, like system theory's, is to continue to be absorbed into routine intellectual, critical and even scientific discourse." [1]

I agree with Benedikt about the immense value and profundity of Derrida's contributions that should not be forgotten as a fad. The baby should not be thrown out with the bath water of stylistic fads. Derrida gives us the critical contraptions that can be employed to systematically dissect a given construct, be it a book or a building or a software package, and we would be well advised to keep the baby, as it were, less the bath water.

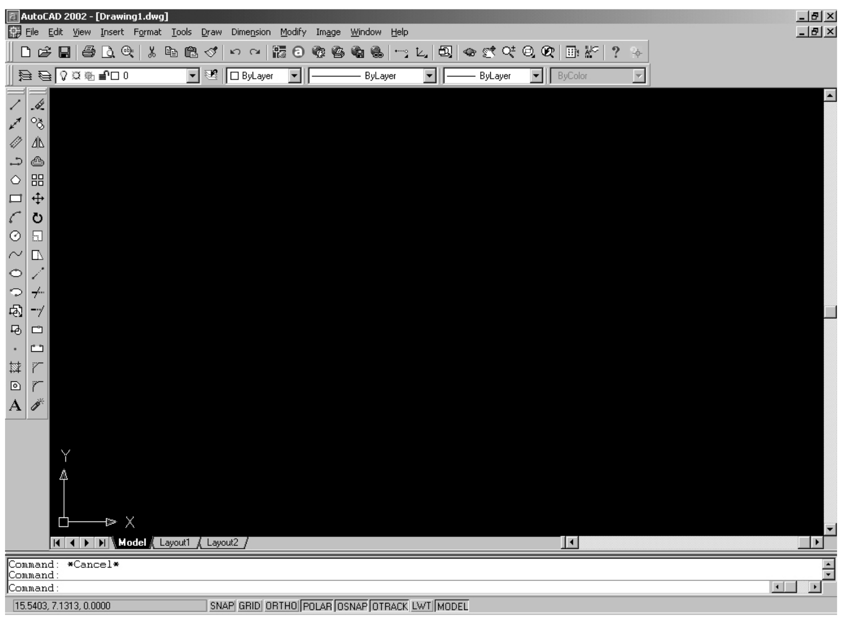
2. Deconstruction of (De)Faults, a Solicitation

"Structure then can be methodically threatened in order to be comprehended more clearly and to reveal not only its supports but also the secret place in which it is neither construction nor ruin but liability. This operation is called (from the Latin) soliciting. In other words, shaking in a way related to the whole..." [4]

Adopting a post-structural approach, the paper will solicit the "(de)faults" of the software as a way of "reading into" the intentions and processes behind the software. More often than not, defaults of software reveal the preferences, priorities, ideological proclivities and a number of other interesting aspects of software design. Defaults are also a way of understanding the underlying assumptions, fault lines, and the "unconscious" of the software. While the defaults can often be changed and customized, in many cases the users do not exercise that freedom either in full or in part. Thus, methodologically 'soliciting' the defaults could yield rich insights into software design. AutoCAD cannot just be judged as good or bad. It cannot just be termed efficient or inefficient. It cannot just be analyzed for its "features" or capabilities. It is not the author's intention to write yet another "review" of AutoCAD. The reader will notice that the "critical close reading" intentionally adapts the key strategy of Derridean "undecidability" all through this article. AutoCAD presents a strange challenge to architects. The author contends that AutoCAD is Derridean in the way it functions. We could possibly go the extent of saying that AutoCAD actually employs (inadvertently, I might add) certain Derridean strategies to question the binary oppositions, assumed boundaries and processes in the process of architectural design. AutoCAD subverts certain polarities unique to architectural design in a Derridean manner. AutoCAD is the architects' pharmakon! One might raise the question of whether the software designers originally intended the software to be the critical tool that I submit that it is. To me, it is less important to dwell on the question of whether the software creators intended this subversion or not. Software takes on a life of its own from the circumstances of its design, its times, politics etc., which are sometimes beyond the authors' control.

2.1. Default: nothing/void/black/blank

Let us picture in our mind, one more time, the three-and-a-half million users diligently staring into the black screen of AutoCAD as it starts up. Many questions arise: what is the conception of space and dimensionality of the black and almost blank void that you are faced with when you open AutoCAD?


AutoCAD 2002 - [Drawing1.dwg]
File Edit View Insert Format Tools Draw Dimension Modify Image Window Help
ByLayer
ByLayer
ByLayer
ByColor
Y
X
Model
Layout1
Layout2
SNAP GRID ORTHO POLAR OSNAP OTRACK LWT MODEL

```
Command: *Cancel*
Command:
Command:
15.5403, 7.1313, 0.0000
```

► Figure 1. Nothing/void/black/blank

The traditional process of design involves a piece of paper or cardboard, a scale and pens or pencils with which to sketch. Among other advantages, this process easily allows architects a relative sense of scale. Architects privilege the use scaled grid(s), which AutoCAD's default black screen subverts. But in AutoCAD, by default, one is faced with a black and blank screen. Conceptually it is a black hole, a *niger tabula rasa*. It is Supreme Nothing. What is the size of the space on that black screen? What tells you whether the space is millimeters or kilometers in magnitude? What scale is it at? What plane are you looking at? How deep is it? What are its boundaries and where are they? These questions cannot be answered without some active intervention by the user. However, even the most avid users remain disoriented and the scale remains undecidable.

Scale is one of the most difficult issues with which to deal while working on a computer. Scale is a matter of relationship between sizes of things and that of the human body. Generally, design software packages employ a variety of ways in which the idea of scale is conveyed to the user. Grids, pixel/vector resolution, visual scales placed around the work area etc. are used to make the user get a better feel of the scale of the project. One could say that it is commonsensical that such provisions are made by the software designers. But, to persist with the black screen interface, the makers of AutoCAD have to be either fully convinced of its ideological value or they are simply too dim-witted to understand simple interface issues. Let us assume that it is the former. Please allow me to elaborate.

The notion of "*nothingness*" has had quite a profound impact on the Eastern as well as Western philosophies. In the Zen and the Vedic traditions, the notion of nothingness holds the highest place. It is at once the emptiness and fullness of being. It *is* and it is *not* at the same time. It is the Silence from which arises the song of the universe. It is everywhere and nowhere. Nothingness is the origin from which everything originates and into which everything dissolves. On the other hand, for existentialists such as Camus, Sartre and Kafka, nothingness is the dreary dead-end of no-meaning: death! It is the "*absurd*." While my intention is not to delve into the notion of nothingness at any great length, I would like to point out the force with which the notion of nothingness has always animated the human mind. It is a condition beyond scale, dimensionality, materiality and attributes. Did the makers of AutoCAD have this intention and thought behind the black screen of AutoCAD? We cannot answer that with any degree of certainty. The maker's logs do not evidence the existence of such thinking in the making of AutoCAD. But, to me, that's beside the point. Ultimately it is undecidable whether the black screen of AutoCAD is evocative of the serenity of Zen tradition or the angst of the existentialist absurdity or if it is just a case of plain dimwittedness.

The black screen essentially stumps any assumptions about scale or size or height or depth in a truly deconstructivist manner. It is a perfect antithesis to the discourse of place, location, orientation and context.

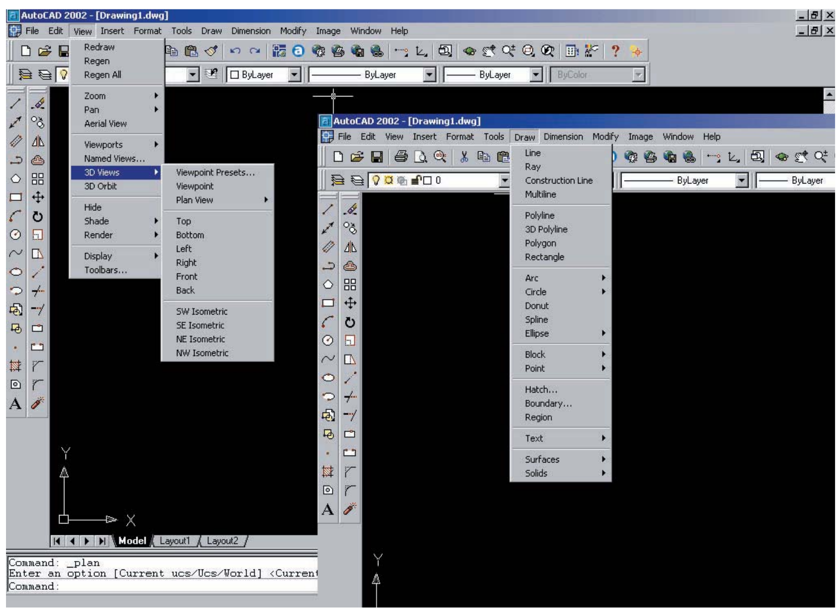
2.2. Default: D, D, D ...

Architects take pride in their three-dimensional thinking. Their processes often involve extensive perspective sketching or 3D model making during early design exploration phase as well as during design development and presentation phases. Not surprisingly, 2D is seen as the polar opposite and often looked down upon by designers. AutoCAD effectively exposes the designers' privileging of 3D by employing a number of strategies:

1. The default screen in AutoCAD is two dimensional (X-Y). It suppresses the third dimension (Z) or the fourth (t).
2. To flip to a three dimensional spatial view, one has to go through a ritual involving three levels of obscure menus and deeply buried icons. Even then one can only arrive at an isometric view, and not a perspective view.
3. Generating a perspective view in AutoCAD is a laborious task and discourages extensive use of 3D perspective visualization.
4. Modeling in 3D in AutoCAD is an awkward, non-native and cumbersome process.

It is very rare to be able to get a glance at the inception and evolution of a software package from the beginning. Please allow me the latitude of taking a little detour and examining AutoCAD's co-founder John Walker's software development logs, which might shed some insightful light [7, 8]. "A three-dimensional capability is desirable. It appears that an ``extrusion" feature might be relatively simple to implement and sufficient for some users. Could

▶ Figure 2: Flatland



```
Command: _plan
Enter an option [Current ucs/Ucs/World] <Curren
Command:
```

be an extra-cost option" [7, 8] Here is the chronology of inclusion of 3D capabilities into AutoCAD: 3D Level 1 Version 2.1 May 1985; 3D Level 2 Version 2.6 April 1987; 3D, full Version 10 October 1988. His logs dated September 5, 1983 also make it clear that 3D was low on AutoCAD's priorities during its development. 3D capability was added only in 1985 in Release 6 (Version 2.1):

> "Many of our competitors (MCS, Nelson Johnson, ESC), have or will be introducing 3D packages around the time of COMDEX. If we do not have a credible response to queries about 3D, we may be in trouble selling our package. While all drafting is 2D, and almost all users will spend all their time with AutoCAD working in 2D mode, 3D is important more from a marketing perception standpoint than a technical one." [8]

Most architects are practical beings. By the word practical I mean limiting one's actions and imagination within the boundaries of what one sees as being possible. Although designers such as Gehry are bent upon expanding the notion of what is practical and possible, in general terms, the notion of practicality is about drawing a very tight circle around one's imagination and action. Why do I mention this? Physicists have been, for well over a century, harping on the existence of far more than four dimensions. Architects privilege three dimensional thinking, often ignoring even the fourth, let alone the dimensions beyond the fourth. I must enunciate and define what is meant by the word *dimension*. The word derives from Latin *dimensio*, which means to measure in between. Dimension is a measure of a phenomenon that cannot otherwise be comprehended. Space as we experience it cannot be explained unless we use at least the three dimensions of length, breadth and height. Without the dimension of time, we cannot explain any change in the way things are. In that sense, physicists have posited that there are phenomena at atomic and universal levels that are unexplainable without the possibility of higher dimensions. Albert Einstein, David Bohm, Stephen Hawking and countless other contemporary physicists have written extensively about seven dimensions and beyond. It might be irrelevant to be aware of these possibilities while putting a brick over a brick. But, the richness of life depends directly on the richness of mind and architecture's or software's ability to reflect/inspire/resonate that richness through concrete work.

Perhaps to point out this dogmatic privileging of 3D, AutoCAD does a Derridean inversion of yet another binarism. It exposes and, more importantly, obfuscates the whole issues of dimensionality (not to be confused with "dimensioning") by reversing the polarity and forcing the 2D upon the designer. Those who are uncritical of this inversion become prisoners of AutoCAD's two-dimensional flatland.

2.3. Default: order/disorder

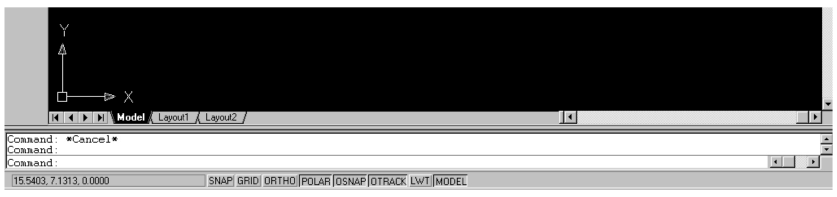
Architects define their role as designers of space. They essentially order space in various ways to achieve their design goals. But, if you go by architects' vision of the world order in general, you will end up with a neatly arranged orthogonal grid or variations thereof. Let us exclude architects of Louis Kahn's caliber who are capable of imagining the widest and wildest definitions of order such as "*Order is*." Except for a select few, the majority of the architects reject any kind of messiness or chaos. They would like to be able to explain how the entrance leads to the lobby leads to the auditorium, etc. Strangely enough, it takes physicists, chemists and artists to recognize higher forms of order in the form of complexity, chaos and fecund messiness. This is true with software design as well.

Now look at AutoCAD's interface (Figure 1). There is a Derridean drive to obfuscate any clear cut organization of tasks via icons and menus. Drawing tools can be found strewn among the icons to the left, among the pull down menus at the top. Window management, layer organization, text management, 3D drawing, viewing, preferences, image management etc. are all dispersed all over the interface in what seems to be a deconstructivist effort to defy a singular comprehension of the interface. The lack of any particular process-oriented order either in the form of workbenches or procedural routines is reflected in the singularly task-oriented icons. AutoCAD's default icons, menus and buttons offer little or no streamlining of tasks and procedures that are often at the heart of any design process. It is assumed that such streamlining or ordering small tasks into a larger, more meaningful procedure is left to the third party plug-in developers. So, what is the sense of order implied by AutoCAD's interface? There is essentially no rule by which one can find a particular feature geographically on the screen. This fact flies in the face of what architects have always known: clear cut spatial/geographical organization. Thus, one more binarism is reversed by AutoCAD's interface.

2.4. Default: drawing/writing

Architects and other designers take pride in their drawing skills. Often the designers are heard promoting how graphic communication is essential to the design process. Also, designers are people of few words. If a designer wants to draw a line, s/he draws a line. If s/he wants to draw a wall, s/he draws a wall. We do not see designers having to first write the word "line," write a verbal description of its color, length and other properties before s/he commits to actually drawing it on paper! Drawing is the privileged/valorized mode of communication and expression among designers.

► Figure 3. "A Command Line Runs Through It"



```
Command: *Cancel*
Command:
Command:
15.5403, 7.1313, 0.0000
```

AutoCAD deconstructs that process by introducing writing as a means to drawing. Command line window is the second most prominent element on the interface after the big black screen. In the tripartite division of AutoCAD's interface, the command line window forms the "base." Command line is where one essentially writes one's way through a drawing. If one is skillful enough, one does not have to even touch the mouse or click on the screen, ever. By placing a great emphasis on writing and by placing the command line prominently on the geography of the interface, AutoCAD inverts the binary opposition of drawing/writing into writing/drawing, thus it challenges the traditional polarities of architecture.

2.5. Default: *Undo* attention

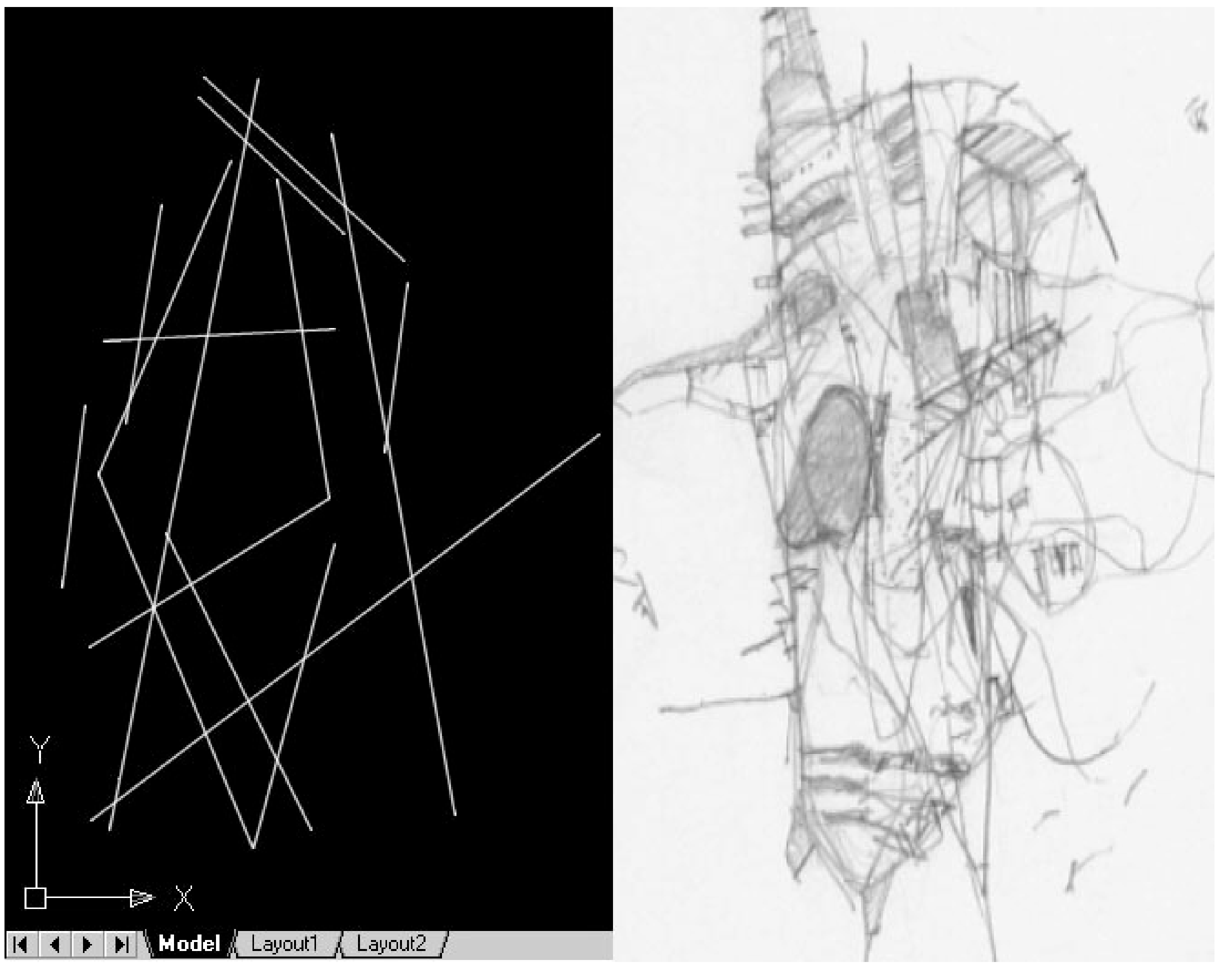
The single most delightful element of any software has got to be the *Undo* feature! At least within the space-time of the computer, the fantasy of time travel and the ability to step back and undo is an indispensable luxury. The metaphysics of undo are fascinating to contemplate. However, I will focus on deconstructing the "time travel issues" in AutoCAD. AutoCAD empowers its users to step back and undo their drawing sins all the way back to the Creation. Although AutoCAD allows the users to undo without limit, once undone, like in a broken time machine, s/he loses the ability to come back to the present! One can redo only by one step forward and no more! This limitation makes us pause and think about the realities of life and time. Many films were made with this theme of time travel: *Planet of the Apes* where the protagonist is thrown back into times when the Apes take over the world, *The Philadelphia Experiment 2* where the Nazis win the World War 2 in a parallel (alternative?) world, etc. Many other films have used this premise to varying degrees of cinematic success. Just by not allowing us to redo endlessly till the present, AutoCAD sets a new swarm of thoughts (and sometimes emotions!) in motion. It can be a really stimulating swarm unless we use AutoCAD uncritically. If used critically, AutoCAD's limited Redo feature comes across as a witty and philosophical move. If used uncritically, it catches one by cruel surprise.

The model of undo implemented in AutoCAD is a linear one, which preserves the arrow of time and action on one axis. However, by limiting the redo, AutoCAD forces the question of linearity, non-linearity and selective branching of time. It raises such questions as, how to undo an undo, how to redo an undo, how to redo a redo and how to capture the flux of action on a timeline for a future (!) use. Particularly, given the abundance of theoretical and practical work on the issue of undo in the computer science and design computing disciplines, one has to ponder the undecidability of the makers' intentions. It makes one wonder! Just as in case of other defaults, it is difficult to attribute such a process of deconstructive thinking to the software makers. But the possibility cannot be excluded.

2.6. Default: ambiguity/accuracy

Architects prefer to maintain a certain level of ambiguity, over-sketching and looseness of lines in order to avoid premature fixation of form and order, particularly during the formative stages and partially during the design development stage. Sketches, schematic drawings and other exploratory drawings clearly show this quality preferred and privileged by architects: grey marks are made against the off white or faded yellow butter paper. Architects are often seduced by the palimpsest-like drawings that contain many layers of direct, indirect, pertinent, impertinent, technical and non-technical notations, erasures, residues, imprints, etc.

AutoCAD thwarts any ambiguity with its ruthless accuracy (or at least the appearance of ruthless accuracy): pure white lines against pure black background. That is a pure binary reversal. Architects might find its accuracy menacing. But the value of this overturning of ambiguity lies in its questioning the habits of the architects. It is the anti-palimpsest. Interestingly, AutoCAD's interface obfuscates what should otherwise be clear and brings to sharp focus what should be otherwise fuzzy. Perhaps this is indeed what Derrida calls an "*aporia*" or an internal and irresolvable contradiction. AutoCAD seems to be ripe with critical intentionality and subversive verve.



▲ Figure 4. Ambiguity/Accuracy

2.7. Default: analog/digital

Architecture is an analog affair. In order for a building to be big (compared to its surroundings), it has to be built big. In order for a building to be narrow, it has to be built narrow. A "thick" wall necessarily has to be drawn and built thickly. Language, on the other hand, is digital. The word "big" is smaller than the word "small." To visualize a red interior, the architect has to use red color markers or pastels or pencils to sketch it out in red. Architects are used to analog modes of visualization and building. It is not in the DNA of architects to be colorblind (albeit some architects maybe literally, medically colorblind). Color, with its power to evoke, provoke, alter the mood and create an ambiance, is a fundamental part of spatial experience. So much so that even musicians speak in terms of the color of sound.

AutoCAD is colorblind by default. It is also lineweight-blind by default. While the "color assigned by layer" facility was available from AutoCAD's first version, "entities having their own color irrespective of the layer on which they reside" was introduced only in AutoCAD Release 7 (V. 2.5). An examination of John Walker's logs reveals the following deliberation in the list of "Questionable Items": "Should entities have colors? Should color be associated with an entity rather than with a layer?"

The concept of "*line weights*" was introduced for the first time in version 2000 (To give this introduction a timeline, AutoCAD Version 2.1 translates to release 6. Here is a list of all releases of AutoCAD till date: *AutoCAD 2004, AutoCAD 2002, AutoCAD 2000i, AutoCAD 2000, Release 14, Release 13, Release 12, Release 11, Release 10, Release 9, Version 2.6 (Release 8), Version 2.5 (Release 7), Version 2.1 (Release 6), Version 2.0 (Release 5), Version 1.4 (Release 4), Version 1.3 (Release 3), Version 1.2 (Release 2), Version 1.0 (Release 1)*) [8].

AutoCAD's lack of emphasis on visual line weights throws the analog architects into a digital world where colors and line weights either do not exist or do not correspond to each other or are achieved and coordinated with great difficulty. By subverting analog thinking, AutoCAD unwittingly does a Derrida on design process. In the hands of a color-aware and visually inclined designer, AutoCAD does not fit like a glove; it wiggles like an alarmed alligator! In the hands of an uncritical user, AutoCAD becomes a digital slow poison that erodes the user's analog faculties and throws the user into a state of undecidable abyss between analog and digital neurons. Used critically, it deconstructively questions the merely analog design thinking. Perhaps AutoCAD is, as Derrida famously pointed out, like Plato's "*Pharmakon!*"

3. Methodological issues for further research

One of the intentions behind writing this paper was to put forth a larger critical framework to understand software (design). For further research, a number of questions could be raised based on the present paper.
What are the aporias (irresolvable internal inconsistencies) fundamental to the functioning of the software?

1. What are the assumptions that the software makes in "constructing" the program?
2. How could one deconstruct software?
3. How could one deconstruct using the software as a critical and/or political tool?
4. Does one look into the text of the code? Or does one look into its mere manifestation when it is "run"?
5. Where exactly does one intersect the hidden agendas of the software?
6. Following Derrida's conclusions that eliminate the notion of

geometrical truth and certainty, how does one deconstruct the geometric modelers?
7. What IS software and what are its boundaries?
8. At what point does the "text" of the software become the "quasi-space" of its run-time self?
9. How about the version of the software?
10. Should one also take into consideration the institution of software making also in the critique?
11. Should software, which is a quasi-spatial construction, be subjected to similar critique as Michael Benedikt did with the Kimball Museum?

More methodological issues: You can deconstruct a book with a book. You can deconstruct a building with a book. But how can one deconstruct a building with a building or a book with a building? More to the point, can you deconstruct a piece of software with another piece of software or, lo and behold, with a building!?

Unlike text, software always actively "does" something. It is *performative* in nature. Text is passive in its existence on paper. So are buildings (unlike machines), with their frozen gestures firmly rooted on site. While a text's meaning lies in its being "read" and a building's meaning lies in its being "inhabited," software's meaning lies in its being "engaged" through action. Strictly speaking, software is a framework for a set of actions: some predictable and others unpredictable. While books can be translated into other languages and still bear the original author's signature, buildings do not have that luxury (Derrida explicitly decries this possibility. For him, translated work is a new text on its own and not a mere copy of the original.). Translation of a software package is expected to result in the same and exact manifestation of its runtime self across a variety of hardware platforms. Software translation usually involves translating the code from one programming language to another and from one hardware platform to another. How does a software package running on LINUX platform differ from its counterpart running on Windows XP®? This presents special problems as to how and what is deconstructed and the ultimate value or legitimacy of those operations. Interesting questions these, but these questions obviously go beyond the immediate goals set forth for this paper.

4. Epilogue

Deconstructing just the interface of AutoCAD has yielded a plethora of critical insights into the software's relationship to architectural design process. In the end, the critical value of AutoCAD depends on how the user unorthodoxly and reflectively engages the software in the design process, which sheds light on the user's own process of design. Much more could be understood if the deeper structures and design of the software are subjected to a critical close reading. It will also be interesting to extend this method of critical inquiry to other software actively used in the design and manufacturing process. It is important for the software designers to go beyond the merely technical discourse of efficiency, performance and tasking to embrace the human implications. Without such a critical discourse, we would end up making servile software that is ethically, culturally and politically specious.

Acknowledgements

A shorter version of this paper was published in the Proceedings of Sociedad Iberoamericana de Gráfica Digital (SIGRADI) conference, Rosario, Argentina, November 2003 under the title "Deconstructing AutoCAD: Toward a Critical Theory of Software (in) Design."

Bibliography

1. Benedikt, Michael, *Deconstructing the Kimbell*, Lumen Books, New York, 1991.
2. Derrida, Jacques, *Edmund Husserl's 'Origin of Geometry': An Introduction*. John P. Leavey, Jr., tr., Hays, New York, 1978.
3. Derrida, Jacques (1976). *Of Grammatology*, Gayatri Chakravorty Spivak, tr., Johns Hopkins University Press, Baltimore, 1976.
4. Derrida, Jacques, *Writing and Difference*, Alan Bass, tr., University of Chicago Press, Chicago, 1978.
5. Norris, Christopher et al (1988). *What is Deconstruction?*, St. Martin's Press, New York, 1988.
6. Strathern, Paul, *Derrida in 90 Minutes*, Ivan R. Dee Publisher, Chicago, 2000.
7. Walker, John, *The Autodesk File: Bits of History, Words of Experience*, Que, Indianapolis, 1989.
8. Walker, John, The Autodesk File at http://www.fourmilab.ch/autofile/www/autofile.html, October 2003.
9. R. Mancini, A. J. Dix and S. Levialdi, "Dealing with Undo," in *Proceedings of Interact '97*, Chapman and Hall, Sydney, Australia, 1997.
10. Chang, W., Woodbury, R., "Undo Reinterpreted," in Klinger, K., ed., *Proceedings of Association for Computer Aided Design in Architecture*, ACADIA, Indianapolis, 2003.
11. http://www.architectureweek.com/2000/0712/tools_2-2.html, October, 2003.
12. http://media.corporate-ir.net/media_files/IROL/11/117861/reports/FACTSHEET.pdf, June 2004.

Mahesh Senagala
University of Texas at San Antonio
School of Architecture
501 W Durango, San Antonio, TX 78207, USA
www.mahesh.org