COMPUTATIONAL DESIGN AND AUTOCAD: READING SOFTWARE AS ORAL HISTORY

Mark J. Clayton
Texas A&M University
Department of Architecture
College Station, TX 77843 USA
mark-clayton@tamu.edu

Abstract
Although the user interface and operation of AutoCAD may seem counter to architectural predilections, the software can be better understood in a historical context of themes of computational design, automated production drawings, and design graphics. Current interest in Building Information Modeling draws upon themes in architectural computing established 40 years ago and expressed in AutoCAD throughout its 25 year history.

1. History, experience and evolution
Because of its dominance of CAD markets for the entire era of the personal computer, AutoCAD is an important topic both from a historical perspective and from a pragmatic one of a student’s future employability. This paper presents a novel framework for learning about architectural computing that links the functions and “look and feel” of AutoCAD to historical events and personal experiences. My approach is to treat AutoCAD as a subject of oral history or an artifact in folk history that has arisen not through conscious intention but from reactions and insights and struggles of people who were doing the best they could. The peculiar commitment of Autodesk to preserve a high degree of backwards compatibility in AutoCAD has produced software that encapsulates the entire evolution of architectural computing. This article is a response to an article written by Mahesh Senegala (2004). His article developed a critique of AutoCAD purportedly by using a “deconstructivist” critical method that divorces the discussion from context. His intention is to strip away false rhetoric and hidden agendas and evaluate the software for what it is. He concluded that AutoCAD is largely subversive of the traditions and practice of architecture due to its context-free “black void” initial display, its reliance upon a command line, its awkward implementation of graphic line qualities, its focus upon 2D representations, and its confusing menu structure that is empty of a design process.

I offer a rebuttal to his criticism by reintroducing the context that he purposefully ignored. The historical record of researchers, the commands within AutoCAD itself, and my personal experiences reveal three themes of architectural computing. AutoCAD can be understood as deriving predominantly from two themes: computational design and architectural production documentation. The software was adapted to incorporate the demands of a third theme, that of design graphics. Senegala’s critique, while purporting to reveal hidden agendas and privileged viewpoints, actually grants legitimacy only to the third theme. By ignoring the historical context, one risks overlooking the richness of concepts and practical power of AutoCAD that has made it among the most successful software products of all time. History, in the forms of written records and personal recollections, enables the occluded and mystifying to become clear and understandable.

2. Computational design
I earned a Master of Architecture degree from UCLA in the mid-1980’s expressly to become knowledgeable in architectural computing. Teachers such as Murray Milne, Lionel March, Bill Mitchell, George Stiny, Robin Liggett, and Charles Moore provided assignments in a topic they called “computational design.” I studied the production of floor plans using the mathematical formulation of shape grammars, the generation of facades using parametric routines written in Pascal, and the development of analytical software such as Solar 5
and Climate Consultant (Stiny 1980; Mitchell, Liggett and Kvan 1987; Milne and Labib 1990). AutoCAD was available at UCLA, but we used Computervision and GDS for CAD studies because the minicomputers that ran that software were considerably more powerful than PC’s. However, the focus was decidedly upon writing code to generate form and pattern as fundamental to architectural computing.

This first historical theme, computational design, can be traced to the origins of architectural computing in the late 1950’s. The invention of “high-level” programming languages such as FORTRAN and LISP in 1957 and 1958 inspired thinkers in all disciplines to contemplate the impact of application software to their particular problems (Spencer 1997). In 1958, architects at Cambridge were already contemplating the application of computers to architectural design, embracing a strict functionalist ethic and dreaming of automated design (Keller 2005). Keller writes “In the work of the Cambridge methodologists, texts, formulae, diagrams and computer code replace plans, elevations and photographs as the proper tools of architectural research; and the long-held understanding of the building as an object of sensory engagement is replaced by the idea of the building as a system of functional relationships …” (Keller 2005, 303). It is important to realize that the birth of architectural computing was driven by passionate functionalism and decidedly non-graphic.

By the early 1970’s, the dreams and hypothetical musings of the earliest researchers were beginning to be implemented. Aerospace corporations developed engineering design software and then sold it on the packaged software market (Campbell-Kelly and Aspray 1996). Although overpriced for all but the largest architecture firms, these packages set precedents and standards for architectural CAD.

Nigel Cross has provided a thorough synopsis of architectural computing research before 1977, and added his own insights and experiments in pitting man against machine (Cross 1977). He lists numerous studies, in space planning, multi-factor evaluation, and other non-graphic design areas. Almost in passing is a mention of computer graphics. The writings of Negroponte from the same era portray a similar landscape of research and ideas (Negroponte 1975). The Oxsys software, perhaps the most ambitions architectural computing effort of the era, is an expression of a similar worldview. It automated the design of hospitals using space layout optimization that referenced a kit of parts (Hoskins 1977). It produced drawings and other documentation for construction and thus was arguably the first “Building Information Modeler”.

Clearly, in the late 1970’s the idea of architectural computing was largely non-graphic and focused upon a functionalist theory of architecture. As the original designers began to conceive AutoCAD, they naturally expected that the users would want programming interfaces and would adopt a computational design approach. AutoCAD, with its LISP programming language and command line, has been an excellent platform for rapid prototyping ideas of form generation, pattern making, shape grammars, and automated production (Schmitt 1988).

Autodesk has been consistently attentive to the needs of developers in commercial enterprises and academia alike. For example, my doctoral research at Stanford University decomposed the design evaluation process and distributed it across the Internet onto multiple machines (Clayton 1998). The AutoCAD Runtime Extension (ARX) allowed me to easily write C programs and load them into a running copy of AutoCAD to allow communication with expert systems, external applications and Web browsers running on separate machines. I appreciated that AutoLISP coupled with Dialog Control Language was a very easy way to build a high quality user interface and install tailored commands into the CAD system. No other CAD system could have supported such an ambition with such ease.

Autodesk has added additional ways of integrating AutoCAD with other software, such as Component-Object Model (COM) interfaces, Visual Basic macro programming, Visual LISP, hyperlink attributes for every AutoCAD entity, and Drawing Web Format (DWF). The company has remained true to the vision of computational
design since its founding in 1982, embracing the product modeling research community in 1994 with its founding of the Industry Alliance for Interoperability (renamed the International Alliance for Interoperability) (IAI 1996).

3. Architectural production documentation

Computer graphics followed its own peculiar trajectory of development. *Sketchpad* is the most famous of early experiments in computer graphics (Sutherland 1963). Sutherland described a computer that combined special hardware and software to enable the user to create drawings and store them. Information sketched could include lines and circular arcs with some simple topology preserved so that vertices could be edited or lines made parallel or objects moved. Lines and arcs could be composed into symbols that could be copied easily. Redefining a symbol would cause all instances of that symbol to be redrawn. The basic ideas are still the foundation of CAD systems.

In the late 1970’s, in spite of the availability of mainframe and minicomputer CAD systems, CAD was not of interest in most architecture firms. Perhaps fostered by new technologies of photocopiers, Mylar drawing film, and more sophisticated diazo printing methods, overlay drafting represented the cutting edge for production of architectural drawings (Stitt 1984; Woods and Powell 1987). In 1980 I was introduced to pin bar drafting and sophisticated reprographics while working in Washington, DC. The blueprinting staff showed me how to print reverse reading sepia mylars, make full size photographic negatives of drafting sheets, composite multiple sheets on a vacuum press, and other esoteric details of overlay drafting. Master draftsmen demonstrated to me dry transfer lettering, drafting templates, transfer hatching, and detail files. Almost all CAD systems, AutoCAD included, are software metaphors for the techniques that I learned in non-digital form as overlay drafting.

Architecture firms in the early 1990’s found that they had invested large sums of money in CAD hardware and software but had realized few gains in productivity. They realized that standardization and the promulgation of best practices was critical to the effectiveness of CAD. The adoption of the AIA CAD Layer Guidelines should be deemed a very important contribution (AIA 1997). The ideas developed by these practitioner-researchers produced insights into the process and techniques of drafting and drawing (Jules 1997; Sanders 1996). Autodesk watched these ideas closely and added commands for external references, paper space, line weights and styles, and layer management.

4. Design graphics

Sketching and rendering, although frequently at polar opposites of the architectural production sequence, are widely accepted as critical design skills. One might refer to them as “design graphics.” Admittedly, AutoCAD in its early forms neglected design graphics, perhaps as a polemical functionalist position or perhaps merely because the available hardware was grossly inadequate. A CHA graphics adapter for an IBM PC of 1983 provided only 4 colors and 320 200 pixel resolution (Norton 1986). Outside the confines of PC hardware of the middle 1980s, researchers worked to refine modeling and rendering algorithms in the belief that realistic perspective imaging was a missing and essential feature for effective CAD (Greenberg 1986).

After the introduction of the Macintosh II in 1987, PC hardware gained the capacity for at least moderately good rendering. The architecture schools began to adopt CAD with at least mild enthusiasm. I helped Cal Poly convert from PC computers running MS-DOS to Mac II’s. Because the popular CAD software (AutoCAD, Microstation, VersaCAD, DataCAD and others) was not available on the Mac and did not support rendering, we used niche software such as DynaPerspective, ArchiCAD, MiniCAD, and Power Draw. The big vendors struggled to catch up, and by 1994 AutoCAD provided surface modeling, Phong shading, solid modeling and ray tracing.

AutoCAD is vulnerable to criticism that its modeling interfaces, line quality, and rendering ability are not “designerly.” Senegala points out these inadequacies in his paper. However, when used in conjunction with
Architectural Desktop and Autodesk Viz, it provides impressive rendering capability and powerful tools for conceptual design. It will even smudge your drawings and add scribbly lines to make plots look hand drawn, if that is important to you.

5. \((CD + APD)/DG = BIM\)

Autodesk remains a leading CAD vendor partly because it watches closely the innovative ideas from the research community and it has the resources to translate them into software. The company’s strategy in recent years reflects a commitment to Building Information Modeling while preserving a migration path for its customers. The strong core values of computational design coupled to the emphasis on automating production drawing is factored by powerful computer graphics imaging to produce a potent modeler. Revit is a step further, providing a cleaner interface and a higher degree of automation of the coordination activities. It largely throws out the overlay drafting metaphor by managing the display of objects in display sets that are linked to the view type. The next step in Building Information Modeling will require the industry to jettison not only the overlay drafting metaphor but even the notion of drawings themselves. A designer could focus on building an accurate 3D, spatial, material, visual model of the building. The users of that model could extract whatever textual reports, 2D views, and 3D views of the model that are required for the tasks of shop drawings, construction supervision, and building operation. My dissertation took a small step along this long research path by distilling the comprehensive 3D model into “interpretations” that filtered the model for the distinct engineering discipline and extracted the mathematical quantities needed to perform cost estimates, spatial program audits, energy simulations, and egress checks (Clayton 1998).

In this vision, drawings are automated derivatives of the BIM, akin to a record set generated by a SQL command rather than a handcrafted artifact. A question for researchers is “what exactly are the reports needed by each of the hundreds (or thousands) of participants in designing, constructing, and operating a building?” Can we formalize the reports sufficiently so that software can derive them from a BIM on demand? We need a Building Information Query Language (BIQL) to complement the BIM.

6. Conclusions

This history of architectural computing, albeit personal and inadequate from a scholarly perspective, provides a context for understanding AutoCAD. Senegala’s critique can be answered with respect to why AutoCAD is the way it is, although one may still object to its idiosyncrasies on grounds of preference, aesthetics, or theoretical purity. A historical review shows that AutoCAD has been in the mainstream of theoretical and practical advances in architectural computing. However, the field itself is shaped by two maverick views about the discipline of architecture. First, the truly revolutionary ideas of computational design have flavored architectural computing research of the past forty-five years. They are reflected in the organic structure of AutoCAD as its command line and programming interfaces, including LISP, ARX, COM, Visual Basic, Visual LISP, and hyperlinks. Second, the industry demanded tools to accelerate the drafting and production phase of architectural projects and committed in the early 1980’s to the overlay drafting metaphor. AutoCAD has largely accepted the metaphor and through the years has implemented successive refinements and expansions within the confines of the metaphor.

Senegala based his criticism on an unspoken assumption that the discipline of architecture is concerned primarily with graphics, and more specifically, sketching and rendering. He identified aspects of AutoCAD that seem counter-intuitive from the filtered perspective of design graphics. While his is a defensible position (it traces back to Brunelleschi’s invention of perspective drawing in the Renaissance), it is arguably out of the mainstream of either practice or research. His views are, however, in the mainstream of magazine architecture, the star-architect system, and much of the professorate that prizes image above everything else. As with much deconstructivist literary criticism, his article, while exposing hidden
agendas and unquestioned assumptions, substitutes other hidden agendas and unquestioned assumptions. AutoCAD and its sibling Autodesk products are marvelous study aids for learning about architectural computing. They can be used as the starting point for introducing nearly all of the major concepts of the field. By employing a historical framework, one can introduce personalities and events to enliven a class and draw students into the intellectual discourse that inspires innovation. It is time for us to realize that there is a history of architectural computing and begin learning from those who went before us.

References


IAI. 1996. IFC Project Model Specifications, version 0.94. Industry Alliance for Interoperability.

Jules, F. 1997. AutoCAD conventions for architects. Albany,


Mark J. Clayton,
Executive Associate Dean, B. Architecture,
Virginia Tech; M. Architecture, UCLA;
PhD in Civil and Environmental Engineering, Stanford University.
Areas of interest: Facility management, design, process modeling, product modeling, sustainability.