INSIDE OUT

Brian R. Johnson
University of Washington
Department of Architecture
208 Gould Hall, JO-20
Seattle, Washington 98195

ABSTRACT

An effort to generate discussion, this paper suggests that between 1980 and 1990 a significant and undesirable change has occurred in academic architectural CAD. We have moved from being developers of ideas and technology on the inside of the development loop to being consumers of products developed in the commercial market place, outside the loop. Certain negative consequences are discussed. Finally, some suggestions are made for turning ourselves "right side out" again.

1.0 YESTERDAY

The year is 1980. Fresh out of graduate school, I've just started teaching "computer graphics" (not yet CAD). The Macintosh with its graphical user interface is still four years in the future, though you can see the Xerox Star at trade shows. The first Apollo workstation, complete with a networked hard disk and real-time graphics, has just been released. The IBM PC is just about to appear. No one's heard of MS-DOS. CP/M dominates the personal computer world, along with UCSD Pascal and various other operating systems. One or two rudimentary drafting systems are available on microcomputers, but AutoCAD is still some time in the future. Movie.BYU is one of the most advanced university-supported visualization products around, and at SIGGRAPH in Seattle the "fractal revolution" is just beginning, with Loren Carpenter's now-famous Vol Libre animation (Carpenter, 1980), among others.

Schools of architecture are either uninvolved in computing or have small but active development projects for software to do 2D drafting, 3D visualization, animation, architectural databases, floor plan layout optimization, structural, solar and lighting analysis, etc. These development efforts usually involve small groups of students and faculty, off to the side of the mainstream of the department.

The standard (and virtually only) book on CAD is Bill Mitchell's 1977 text Computer Aided Architectural Design, a snapshot of the academic, intellectual approach to computing in architecture. This book addresses the essentials of a 2D or 3D drawing system, transformation matrices, primitives, and object-oriented display lists, but it goes far beyond this. Large sections of this text discuss design analysis, data structure, optimization, simulation, and so on. This text looks at CAD from the point of view of educating professional architects who have an interest in advancing the application of computing techniques to architectural settings.
2.0 TODAY

By contrast, in 1990 there are dozens of books about CAD. Readers and publishing houses both realize that there is much to be said about this subject. However, many of the new books are efforts to augment the documentation provided by software vendors, only a few are general texts. Even among these, Mark Croxley's representative book, *The Architect's Guide to Computer-Aided Design* (Croxley, 1988) is mostly about fundamental CAD Drafting concepts and "how tos" (though he does an admirable job of surveying some of the more advanced topics).

All of the attention would seem like a dream come true to many of us a decade ago. Unfortunately, in the intervening decade the popular definition of "CAD" has changed. Now, rather than receiving an "insider" treatment built up from conceptual foundations and empowering critical review, it has become an "outsider" topic, treated with feature lists and glitzy promos. I think Croxley's is one of the best of the recent crop, but readers of his book are clearly not on their way to becoming developers of CAD systems, whereas readers of Mitchell's book might have been.

The explosion of interest has caught universities a bit flat footed. Students are actually seeking more exposure to CAD. Many programs have been scrambling to satisfy the demand, with new faculty positions and courses. Often these courses focus on operation of commercial microcomputer CAD programs (after all, it is difficult to learn about a topic you have not experienced, and heaven knows, it's easy to make a quarter or semester course out of x-CAD). Unfortunately, such courses are frequently called "computer literacy" courses (probably in hopes of satisfying the NAAB requirement) and frequently students do not receive much additional exposure. Fewer and fewer schools (as a percentage if not in absolute terms) maintain a developmental line of endeavor. Software is acquired from the commercial market and the focus is on application of the software.

3.0 GETTING "INSIDE OUT"

There's no question that two dimensional CAD systems are very successful commercial products. Why might this be? What set of factors has contributed to the success of this software over others? It isn't that other's didn't exist: MOVIE.BYU offered 3D color visualization with surface smoothing in 1980. Systems such as BDS & RUCAPS (both from the UK) offered full "Building Modeling" approaches to CAD, in which the notion of a "drawing as a data extraction on the design" was actually (if incompletely) realized. Why didn't these applications succeed instead of the 2D drafting application? I suggest it is because of the relatively low-powered PC platforms that were available and whose minimal capitalization requirements could be justified in architecture firms. Drafting was a sufficiently undemanding application for these machines, but one which designers with little computer experience could readily understand. Further, many professions draw things, but only architects design buildings, making drawing programs better marketing choices for software vendors. In the end, 2D drafting triumphed over other applications not because of its greater intrinsic value, but because it simply fit the financial and conceptual needs of the industry in the US better than other applications (the results of this struggle have been somewhat different in Europe).

Today microcomputer-based systems are common throughout the profession. They are used predominantly for the production of working drawings and, to a lesser extent, design development drawings. Almost all of this application is in a "pure" 2D form. Installing CAD systems in offices has generated a need for operator training. In part because the profession has no tradition of "in service" training, and in part because drafting offers the entry-level opportunity for young architects, students about to enter the job market feel a distinct need for CAD Drafting skills. They put pressure on their schools for training in operation of CAD systems. They are mostly interested
in marketable CAD skills, not general CAD knowledge. Naturally, most institutions have responded in some fashion: schools with courses, publishers with books.

This condition, of looking at CAD as consumers of the technology rather than creators or theoreticians, is what I call the condition of being "Inside Out." I believe it represents a dangerous cul-de-sac from which we must quickly emerge if the evolution of architectural computing is not to become sidetracked.

Software developers always recognize that other approaches are possible, but naive users of commercial software often define the application domain in terms of their first application.

4.0 IS THIS RIGHT SIDE OUT (WHAT'S WRONG WITH SUCCESS)?

Many CADvocates would argue that the scenario just presented is a description of CAD's success. In many ways it is, or is at least an important part of success, for without a large base of reasonably CAD literate architects it is unlikely that adoption of more adventurous and revolutionary applications will take place. But, are we actually generating such a base? I raise three questions as tests:

1. Are students becoming computer literate?
2. Are students becoming CAD literate?
3. Does the software we are using challenge students to think about the future?

4.1 ARE STUDENTS BECOMING COMPUTER LITERATE

"Computer literate" graduates is one goal for most of our programs, and is expected as a part of accreditation reviews, but is not well defined. One observer has noted, "Computer access does not guarantee computer literacy," and "Students don't become computer literate by learning enough Word Perfect to write a term paper." (Danielle, 1990). Substitute AutoCAD for WordPerfect and this becomes an indictment of many "computer literacy" courses.

Computer literacy describes an awareness of how computers process information, what kinds of information they can process, and what the information "costs" are associated with this processing. A computer literate person recognizes characteristics of programs and is able to use this recognition to anticipate what the program is able to do. To achieve this level of literacy requires exposure to a wide variety of computing tools and explicit efforts to draw together the common threads and highlight the unique qualities of such programs. It requires thinking about computing at a level above individual applications and commands.

By offering up "CAD operation" as a definition of "computer literacy," we are supporting the notion that computers are simply the functional equivalent of a drawing board. However, because the technology of manual drawing is so very different from the technology of CAD drawing, students see them as competing techniques. This reinforces the artificial division between the "designer" and "CAD operator". In the end, it is more likely to be the designers who become principles, and therefore end up making software-adoption decisions, rather than the students we reach with computer literacy courses.
4.2 ARE STUDENTS BECOMING CAD LITERATE

"CAD literate" is even less well defined than computer literate, but it certainly includes exposure to more than one CAD application environment, and must be based on a general computer literacy. I believe that real CAD literacy must be based on exposure to as wide a variety of systems as possible. Most mono-system operators DO NOT understand CAD, any more than BASIC programmers understood "computing." Exposure to a single, albeit rich, application cannot lead to fuller understanding of the medium. It does lead to a variety of "experiential myopia" (the tendency to define the possible in terms of past experience). Each new program expands the envelope of possibility and, through reflection, increases understanding of the systems already used, in much the same way learning a new spoken language enhances understanding of one's mother tongue.

Minds grow on the interplay of ideas and the challenges they pose. Too much homogeneity leads to sterility. By fostering or passively allowing a definition of CAD as "current CAD products" we are mis-educating our students. I've had students come to my office looking for course information who actually believe that AutoCAD is CAD—that that's all there is to CAD. It's like believing that PostModern is all there is to architectural style! They are young, but they are predisposed by an uncritical media to believe, sometimes adamantly, that the commercial products necessarily represent the "best" of CAD ideas.

Understanding current CAD systems in terms of the history of the term and the way it is used in the various industries and professions helps students to see trends and forces within the industry. This equips them to better predict and understand the directions that this development is taking, and where it might go in the future.

4.3 ARE STUDENTS BEING CHALLENGED (OF BIRDS, LOCOMOTIVES, AND AIRPLANES)

The typical CAD workstation for most schools and offices is a PC. These systems are isolated, both because the personal computer file system is fundamentally "stand alone", and because drawing files in most systems exist as largely self-contained documents. This isolation limits the synergy that might develop amongst different disciplines or individuals within the same office or school. When used in groups, they actually diffuse energy by "borrowing" problems from the "paper" world like "Who's got drawing A17?" and adding particularly electronic problems as well: "You mean you both did work on separate copies of the drawing?" Are these the kinds of challenges we want to present?

Even more fundamentally, I am hardly the first to point out that the 2D display-list drafting engine at the core of most commercial products is near it's end (e.g., Eastman, 1989). The very MIPS which make animations possible on "personal computers" will blast away at this software to reveal an informational paucity of majestic proportions. Ultimately, this information dearth puts a very low upper bound on productivity gains which can be realized with these systems.

Just how are these systems "information-poor"? One of the basic problems which drafting systems share is "redundant geometry". The need to create and modify both the plan an elevation drawings of a building (as well as perhaps a separate 3D model) contributes to an update-nightmare.

Still, the task of adding greater "intelligence" to 2D or 3D systems is daunting. A list of graphic primitives and their controlling parameters tells you virtually nothing interesting about a building, and that's what most CAD data bases are. Take such a list and, without drawing the graphic, try to discover certain simple answers: do the objects form one or more complete room boundaries?
given an arbitrary point, is that point inside one of those spaces? are the spaces symmetrically disposed? there are numerous interesting architectural questions which simply cannot be answered with this kind of database (e.g., does the space have visual privacy? does the space have the required number of fire exits? does my data represent an enclosed volume or am i heating the great out-of-doors?).

a good friend and insightful software engineer once observed that, "you can't make an airplane by strapping pigeons to a locomotive." if you start with a systems that wants to model drawings it is very difficult to make it model buildings. even a system that believes in 3d geometries has certain difficulties representing buildings adequately, because such a large proportion of architectural drawing content is symbolic, not geometric.

two-dimensional drawing systems permitted both geometry and symbol to be treated in the same way, which certainly contributed to their success. separate 3d geometry modelers enhance visualization and presentation capabilities, but cannot generate finished working drawings. the conceptual bases of most of these systems require very little time to present, and mostly involve implementing or enhancing techniques borrowed from the 'paper' world. operation of the systems may be challenging, but conceptualization of them rarely is. soon architects are going to master the productivity that they can gain from their existing information systems and are going to demand more. if we have not "salted" the field with some alternative ideas for them to (re)discover, it seems possible that they will conclude (as they did in the '60s) that computing just cannot deliver on its promise.

5.0 getting right-side out

obviously, i conclude that we are, but ought not to remain, "inside out." i think current conditions could become self-reinforcing, leading to stagnation. one article republished in our own newsletter actually says, "it is recommended, therefore, that a/e's concentrate more on preserving their tremendous investment in cadd databases and employee training than on investing in specific product features." (korte, 1989) he goes on to indicate that "cadd packages [will become] less and less distinguishable by product features." i think this means these systems are "topping out."

the keynote address at siggraph this year was given by robert sproull and was entitled "parts of the frontier are hard to move" (sproull, 1990). he states, in part (my italics),

what remains hard is modeling. the structure inherent in three-dimensional models is difficult for people to grasp and difficult too for user interfaces to reveal and manipulate.

only the determined model three-dimensional objects, and they rarely invent a shape at the computer, but only record a shape so that analysis of manufacturing can proceed. the grand challenges to three-dimensional graphics are to make simple modeling easy and to make complex modeling accessible to far more people.

we need to move some of that frontier, and we need to address parts of it that go beyond simple 3d modeling to the modeling of buildings as complete thermal, structural, aesthetic, lighted, acoustic, cultural, and bureaucratic events with human designers and users. i think some of the research done at acadia schools does advance the frontier, but it needs a more visible presence, and it needs to be more accessible to others, both as ideas and as software.
5.1 CURRICULA

We need ideas. We need bright students to help contribute to the expansion of knowledge about architecture and its relationship to computing. Our courses should give all a good grounding, and should challenge some to investigate further. Some of the issues involve computer programming, but many involve simply careful critical thinking. Some questions need someone who will look closely and objectively at functioning offices (and schools). In all cases, the results of these investigations need to be distilled and mulled over, and then brought back to the academic community through papers and articles in whatever fora can be be discovered, including ACADIA.

I believe that a renewed commitment to a broadly based computer literacy goal is appropriate, but can not be built on learning to operate a CAD system. CAD system operation is a desirable skill, but perhaps it should even be de-coupled from CAD theory courses to avoid giving the impression that "existing" and "possible" are one and the same. Separation would serve the majority of students whose interest is in the existing systems, while preserving an opportunity for those who become interested to study more deeply.

Besides making these changes to curriculum, we need to actively seek out and seek answers to interesting questions about computer applications to architecture. Some of these questions have been around a long time, and have proven intractable. Perhaps they can be recast in a way which would permit us to discover why they are so difficult. This list is certainly partial and not especially important. Enhanced inquiry and investigation is.

5.2 RESEARCH

With few exceptions, the hottest architectural applications available today (building modeling, rendering, solid-modeling, etc.) were all available in developed or nascent forms in 1980. They didn't run on PCs, but they existed. The next generation of ideas probably exist on larger systems today, but most schools have invested heavily in PCs. This is a problem. The prospect of "extracting" traditional drawings from the building design database was discussed in Mitchell's 1977 book and again just last year (Eastman, 1989). It has been available to CAD users in the US since the early part of this decade. Of course, it doesn't make sense to run such software on stand-alone PCs because if its requirement of a single database (a fact which has all but killed off the product in the US), but networked databases re-open the possibility of its emergence in the commercial market. How can we bring these new ideas to more students? Academia must work toward finding and displaying new directions for computing to pursue in architecture. Perhaps a consortia approach similar to that of the Canadian schools (Dunahy, 1989) would work here?

6.0 CONCLUSIONS

Practitioners, students, and schools have been seduced by the challenge of learning to use complex commercially packaged 2D and 3D drawing systems on personal computers. This has lead to an inversion of the relationship between the generators of ideas and the consumers, with academic institutions becoming the consumers. I have suggested several ways in which this is not in our best interest, nor in the best interests of our students. I believe a re-dedication to the active exploration of ideas, to tinkering with the systems we have, and generating new ways of looking at the problems we address is essential. I also believe that we should undertake to disseminate those ideas more widely, and that we must take care that our students are exposed to as many ways of looking at the world as possible.
REFERENCES


Danielle, Diane, "Imagination is the First Step Toward Computer Literacy", PC Week, Vol 7, no. 14, April 9, 1990, p 52.


Index of Authors
INDEX OF AUTHORS


