The Millennium Futures to Come
VISIONARY PROJECTS BY NINE AMERICAN ARCHITECTURE FIRMS

Special Section: Product Reports 1999
Here is a crystal ball that can show us the future of architectural software. It depends not on gimmickry but on the fact that tomorrow’s technology goes through years, sometimes decades, of development before it becomes commercially available. All over the world, architecture professors and their graduate students are engaged in innovative software research and experimentation. For many, the goal is to produce inspiring design tools, such as those that make 3-D modeling more intuitive, in contrast to the production tools offered by most commercial software developers. For others, the goal is to improve the integration between applications, promising efficiency benefits to the entire construction industry.

Examples of student research-turned-product include the conceptual modeler DesignWorkshop, from Artifice Inc., which architect Kevin Matthews (matthews@artifice.com) began developing as a master’s thesis at the University of California at Berkeley and further developed while teaching at the University of Oregon. Lightscape rendering software has its roots in Cornell University’s Program of Computer Graphics. Countless other pieces of commercial software have their theoretical or computational origins in the volumes of academic journals from the last several decades.

Unfortunately, it takes more than a good idea to make a marketable product. According to Matthews, the obstacles are both technical and institutional. In academia, he says, a narrowly focused solution is acceptable as a “proof of concept,” in other words, the concept is valid, so fully developing it is unnecessary. Direct interaction between researchers and users makes manuals and technical support unnecessary.

“But to succeed in the marketplace, software has to be part of a complete solution for problems in real-world jobs,” Matthews says. “That means you have to develop flawless software plus accessory information, documentation, training materials, packaging, delivery systems, marketing, sales, and support.” Although professors can receive academic kudos for generating good ideas, they are less likely to be rewarded for all the work required to bring a program idea to market. Furthermore, work that’s done in a university setting is subject to disputes over copyright or patent ownership—does the university itself or do the students and professors retain these rights?

Even so, university researchers tirelessly pursue their innovative work, assuming that the obstacles can be overcome. The following glimpse of four current doctoral dissertations shows some tools that practitioners may be using in the future. Whether all of these will be successfully brought to market is a matter of speculation and luck.
Design by physics
In the architecture department at Texas A&M University in College Station, Tex., doctoral student Scott Arvin (arvin@viz.tamu.edu) is developing a system for "physically based space planning." In other words, the computer prototype accepts building program parameters, such as square footages and adjacency and separation requirements, and uses them to construct a range of viable floor plans. It is the digital equivalent of moving around scaled pieces of paper to create plan configurations, except that the individual pieces can change shape, many complex considerations can be simulated at the same time, and the resulting footprint has exterior walls that are logically aligned.

IF HUMANS CAN INFER DESIGN INTENT FROM SKETCHES, COMPUTERS CAN, TOO.

Imagine that each space in a bubble diagram is attached to other spaces by springs. The architect assigns a relative strength to each spring proportionate to the need for adjacency between the two spaces. Arvin's computer program applies the laws of physics to the springs to pull some spaces together and push others apart, until the configuration reaches equilibrium. Similar physical simulations are performed for moving spaces that require particular views to particular orientations, for moving interior spaces toward the center of the overall footprint, for aligning the boundaries of adjacent spaces, and for maintaining the necessary area or proportion of each space. For example, a kitchen and dining room would be pulled together while a concert hall and a noisy loading dock would be pushed apart. All of this occurs in an animated format, allowing the designers to observe the effects of the parameters they specify.

What makes the software a design tool, one that could become part of a larger design system rather than an exercise in physics, is that it allows the designer to interact with the various parameters and make continuous adjustments. This adds the designer's intelligence to the simulation and affords multiple plan configurations from which to choose. "This interactivity," Arvin says, "evokes the feeling that one is working with a living design, one that responds to the user in ways consistent with the programmatic objectives, though it still provides intuitive designer control."

Right tool at the right time
Many architects remember a favorite design professor who could, after merely glancing at a drawing, pull down the perfect reference book to help develop the idea. If humans can infer design intent from sketches, maybe computers can, too. So reasoned Ellen Yi-Luen Do, (ellendo@u.washington.edu) now a professor at the University of Washington. For her recently completed doctoral dissertation from the Georgia Institute of Technology, she developed a prototype sketch environment in which the computer software recognizes drawn shapes, determines the drawing type, and interprets symbols to derive design intent. It then launches other software applications that perform reference searches or provide some analysis of the drawing.

For example, a certain configuration of lines can be construed to be a floor plan. If the architect draws a few arrows emanating from a point, Do's software infers that he or she is thinking about views within the plan. This launches a program called Isovist, which highlights the portion of the plan visible from the viewpoint, taking into consideration walls, windows, and partitions. The architect then continues designing without having to think about software mechanics.

Do's prototype, dubbed "The Right Tool at the Right Time," or RT², depends on a foundation of sketch recognition software which, like Isovist, was developed at the University of Colorado's Sundance Laboratory for Computing in Design and Planning. RT² can identify whether a drawing is a bubble diagram, floor plan, section, or 3-D view; it can recognize commonly understood symbols such as windows, walls, ground lines, sun angles, and numbers. In addition to Isovist, RT² can launch Archie, a library of post-occupancy evaluation case studies developed at Georgia Tech; the Great Buildings Collection from Artifice Inc, an encyclopedia of hundreds of famous buildings; a numeric calculator; and a geometric modeler that converts sketched rectilinear massing into a
NEW SOFTWARE RELIES ON NATURAL HAND GESTURES TO CREATE 3-D MODELS

3-D model can be turned and viewed from other perspectives. In theory, if a 3-D model can be turned and viewed from other perspectives, this software can be used to manipulate and display the model. 3-D software is used in the tool that was originally developed by 3-D software company, 3-D Systems. The software is designed to work with a variety of 3-D printers, including those used in the construction industry. The software allows users to create 3-D models by manipulating the 3-D printer's print bed to create the desired shape. The software also allows users to print multiple copies of the model and to print the model in a variety of materials, such as plastic or metal. The software is designed to be easy to use and to be compatible with a variety of 3-D printers. The software is available for download from the 3-D Systems website.
performance data from the Department of Energy and ASHRAE to specify wall insulation levels even before the wall material has been selected. Thus, the architect can look at high-level performance evaluations even during schematic design. Moreover, the BDA supports different building assemblies simultaneously so that the architect can compare these configurations—wood studs placed at 16 or at 24 inches on center, for example.

The BDA sports its own Schematic Graphic Editor, which allows a designer to input and edit a simple sketch. But Papamichael expects very soon to replace this with links to commercially available object-oriented CAD systems, such as Autodesk’s new Architectural Desktop. In the future, the BDA will link to any number of architectural applications, such as cost estimating, environmental impact analysis, and electronic product catalogs.

**Putting it all together**

These four examples are isolated approaches to architectural design computing. Each demonstrates a narrowly focused capability that is desirable but missing from current practice. However, each prototype is founded in the belief—ubiquitous within academia, it seems—that the ultimate goal is to connect all tools of drawing, design, analysis, simulation, and visualization to a single, cohesive, complete model that contains everything that can be known about the building under design.

Bringing these ideas into the commercial software market is made even more difficult by this singular approach, which represents a mismatch between the structure of the construction industry and what is needed for the single building model approach to be accepted, says Robert Aish (Robert.Aish@Bentley.com), a research scientist with software vendor Bentley Systems.

Fifteen years ago, Aish worked on an English design system called RUCAPS, which later came to the U.S. and was renamed Aias Sonata. The system allowed multiple users from all disciplines, or “enterprises,” within the construction industry, to work on a single model with parametrically defined components. Drawings are generated from the model, thus guaranteeing their consistency.

Sonata was a commercial disappointment. One reason, Aish says, is that its concept of “enterprise computing” required a restructuring of teams within the industry. For example, it required more detailed input by architects for the benefit of the engineers and contractors downstream. But there was no motivation or compensation for the increased workload upstream. As a result, the now-familiar drafting systems that run on low-cost personal computers won over the U.S. market because drawing efficiency could be improved without requiring major readjustments in the way business is done.

Now software developers are joining academic researchers in calling for the professions to make these adjustments, revamping the relationships between architect, engineer, builder, and owner and rethinking how various players are compensated for their contributions. Until that happens, Aish says, Bentley is working on methods for easing the transition. Their new ProjectBank technology creates a central data repository that looks much like a traditional CAD environment, but is accessible to the entire team.

At the same time, software developers, including Bentley, are working with architectural and sociological researchers who study collaboration. Their goal is to understand the key social and technical barriers that have prevented a restructuring of the design-to-construction process. This time, Aish says, “we want to get both the technology and the business conditions correct—and correctly matched.”

This adds an optimistic note to the story of architectural software research. With sufficiently powerful hardware and support software, all the clever research ideas—automating space planning, hand-crafting forms, calling up reference materials with sketches, and performing complex energy simulations early in design—will eventually become part of a practitioner’s everyday toolkit. The obstacles may be many but the rewards are compelling.