A MACINTOSH DESIGN STUDIO

G.Z. Brown and Barbara-Jo Novitski
Department of Architecture
University of Oregon
Eugene, Oregon 97403

ABSTRACT

During the past year at the University of Oregon, we have conducted an experimental design studio in which each student had an Apple Macintosh SE microcomputer on his or her studio desk. Each term we experimented with a variety of software, furniture arrangements, and pedagogical approaches to integrating computers in design teaching. Like most others who have conducted such experiments, we encountered problems in trying to use hardware and software which is fundamentally inappropriate for the intuitive, graphic, and creative processes characteristic of preliminary design. However, we solved many of these problems and have produced useful techniques that may form the beginnings of a new approach to the use of computers in architecture schools.

Our results fall in three major categories: 1) pedagogical discoveries about learning to design with a computer, which is greater than the sum of learning to design and learning about computers; 2) design exercises based on the Macintosh environment, exploiting the unique graphic qualities of the machine while simultaneously developing the ideas and drawing skills needed in the preliminary stages of design; 3) descriptions of the studio environment, including hardware, software, workstation layouts, security solutions, and other practical information that might be useful to others who are contemplating a similar project.

INTRODUCTION

We are concerned with the creative process of design and how it is affected by the powerful new tool which is invading the architecture curriculum -- the microcomputer. Does this tool empower or constrain the designer? Does it focus our attention on design quality or distract us from it? We have been studying these questions in the context of the studio, the heart of most professional education, where students form the foundations for the character and quality of what they will design later in their professional lives.

As a result of the recent profusion of computers and the cost effectiveness of computer aided drafting (CAD) software for architectural firms, universities and professionals have pressured schools of architecture to integrate computers into the curriculum. Studios have experienced a change of character to match the capabilities of the computer, yet the technology has changed little to meet the needs of studio. While computers have been used effectively in some technical courses, they tend to be the bully of the studio -- powerfully able to command attention, but largely destructive of the creative impulse in design.

Architectural design is a difficult reiterative process: One begins with the conceptual design of a trial object, evaluates it, redesigns it, develops it, re-evaluates it, and so on. In this process, the designer must synthesize a vast array of widely varying quantitative and qualitative information. The challenge is to make computers, which are analytic, compatible with problem-solving

techniques, like architectural design, which are reiterative, graphic, and synthetic.

The brain has been characterized as having two distinctly different modes of thinking [Edwards]. One mode is the rational: it controls the verbal, analytical, objective, and sequenced ways of thinking. The other mode, by contrast, governs the non-verbal, synthetic, subjective, and metaphoric ways of thinking. The one mode is good at figuring things out and making judgments; the other is good at seeing new patterns and spatial relationships and suspending judgment while imaginative leaps are made. While both sets of characteristics are important in the development of creative ideas, modern society and education tend to cultivate the dominance of the verbal, rational mode.

We belive that the education and practice of architecture need bi-modal thinking. An individual's design process ideally makes frequent jumps between the wildly imaginative and realistically sober. Spatial relationships and creative combinations of patterns can only be conceived when the pattern-seeking brain is free to play unencumbered. But these ideas can only evolve into a building design after being subjected to the analytical scrutiny of the rational brain and its concerns for structural soundness, social needs, economies of materials, etc. [Brown and Novitski].

THE PROBLEM OF COMPUTERS IN DESIGN

The computer, with few exceptions, performs best in support of the modes of the rational brain. Typical computer input, even graphic input, is explicit and meticulously defined. In a common scenario, an analysis program demands the dimensions of a building in a sequence inspired by hierarchies of algorithms and the logic of analysis rather than heuristics of human thought. This does not encourage an atmosphere of imagination or experimentation. Drawing programs contribute to this clean, meticulous atmosphere by encouraging a precision that destroys the provocative, sensuous ambiguity of drawings that is appropriate for preliminary design. Computers exacerbate the dominance of rational over imaginative thinking in architecture schools by encouraging students to think that the tightening constraints of a sophisticated analysis or increasing precision will lead to an optimal design solution. In fact, a thorough analysis can only tell you what *not* to do; it can't tell you what *to* do. And precision in a sketch frequently reduces the opportunity for multiple interpretations and the creation of mood and ambiance.

In Nicolas Negroponte's 1975 classic discussion of computers, *Soft Architecture Machines*, he raises many of these concerns. "It is crippling to force an explicitness in contexts where the participant's equivocations are part of the function of design; ... the tedium of overt, categorical exchange is counter-productive, unfulfilling for the speaker, and boring; ... constructive and exciting responses are often generated by twists in meaning that result from the personal interpretation of intentions and implications." It is clear that far more needs to be done to bring computers in line with the way designers think.

A POSSIBLE SOLUTION

We at Oregon have taken an activist position in regard to the role of computers in studio. We feel that computers can, in fact, *empower* the designer by broadening the range and sophistication of the elements that he or she can consider at one time. This improves the designer's ability to simulate the physical, social, economic, and aesthetic forces on a design in a way which reveals their formative character. It also enhances the designer's creative ability by assisting him or her to generate alternate design ideas. Thus empowered, the designer has a more sophisticated lens to focus on the quality of design and perhaps more time to dedicate to it.

We have chosen to use available software and inexpensive hardware rather than to invent new software for state-of-the-art hardware. Other investigators are pursuing this latter position, and it is important that they do so. However, this a very expensive and time-consuming pursuit,

beyond the resources of most schools of architecture. Furthermore, computers are arriving in design studios *now*. It is important that they be used immediately in a way that fosters design excellence. If we wait until the "ideal architecture machine" comes along, our negligence may foster a decline in design quality.

Although computers have been in architecture schools for some time, several recent developments have made them more capable of improving the quality of the design. One of these has been the introduction of the Apple Macintosh microcomputer with three key characteristics: its low cost puts it into the affordable range of architecture school budgets; its high resolution graphics make it capable of reasonably realistic and attractive visual expression; and the graphic user interface makes it accessible and provocative to non-technical, visually-oriented users. Recently other computer systems have begun to emulate these three features.

WHY COMPUTERS HAVE FAILED IN STUDIOS

In general, computers have been treated like group tools rather than personal tools like T-squares, triangles and scales. They are usually housed in a separate room or even in a separate building, away from the design studio. This means that students must interrupt their design process in order to use them, reinforcing the idea that computers are not really central to design. Any designer-to-machine ratio of greater than one means that a computer isn't always available when needed, "when the idea strikes". It also encourages instructors to assign well-defined, non-exploratory problems because they fit better into fixed time slots. The result is that computers are not well integrated into the studio routine and are not seen as indispensable to the problem solving approach. As an analogy, it is easy to see how, if ten designers were forced to share a parallel rule that was housed ten minutes away, only available for a few hours per day, and frequently in disrepair, most would quickly learn how to get along without one.

Another reason that computers have not fared better in studios is that software has developed on a piecemeal basis, focusing on specific tasks like word processing and sketching rather than on complete curricula which are a series of connected problems. So while excellent software exists, one piece tends to be unrelated to the next -- there isn't any connecting tissue. Good programs that do exist are frequently not fine-tuned for architectural use. For example, drafting programs encourage users to think of graphic representations as collections of lines rather than as holistic images.

Finally, architectural studio teaching is steeped in tradition and mystique. Studios are frequently built around the personalities of individual instructors. This means that design curricula change slowly and are resistant to innovation. As a result, studio teaching methods haven't taken advantage of particular capabilities of the computer. Instructors are much more likely to complain about the inappropriateness of professional software than to think about innovative ways to exploit the simple and available commercial applications. They need to see a model curriculum that enhances, rather than threatens, their personal teaching styles.

EXPERIENCE OF OTHERS

Although computers have been used as analytical tools in architecture schools for several decades, efforts to integrate them into the studio curriculum have been relatively few and relatively recent. Variations in these experiments have included the sophistication of the hardware, the ratio of students to machines, the distance between machine room and studio room, the academic pressure on the students to use the equipment, the particular software selected, and the degree to which explicit exercises are developed to demonstrate the use of the computer in particular phases of the design process.

Much of the available literature reports on problems as well as successes. These problems range

from the arcane difficulties of learning a complex technology to the more difficult philosophical questions associated with working in a new conceptual framework that has evolved outside of the tradition of architectural education [J.L. Brown]. Ways the computers have been used range from evaluation of designs via economic, analytic, and other objective criteria to the development of designs based on rules from synthetic shape grammars [Flemming and Schmitt] and visualization of three dimensional and animated color models [Goldman and Zdepski]. While new computer methods for assisting the peripheral tasks in design are developing every year, a controversy has developed over "... the computability of the inherently human process of design" [Harfmann, Swerdloff, and Kalay]. Some researchers have studied the roots of the design cognition in order to understand how the technology might assist rather than impede creative processes [Herbert]. Recently, attention has been paid to ways the computer can help at the beginning at the design process, where studios typically concentrate, rather than at the end, where computers typically excel [Miller].

Many of these investigators conclude that hardware and software technology has not yet developed to the point that there can be true Computer-Aided Design. We at Oregon take a somewhat different approach and ask instead what we can do now in our design courses with the available technology.

THE STUDIO

We began this project in the fall of 1987 with an equipment grant from Apple Computer Inc., which enabled us to create a studio with thirteen student and two instructor work stations. Since then we have received additional hardware and software grants from other companies. With support from our school and department, we have remodeled a studio, installed appropriate networks, and built special furniture.

Our goal was to create an "electronic studio", an otherwise traditional architectural design studio, in which every student had his or her "own" Macintosh computer on the drawing board. By dedicating a machine to each student, we have created a hardware environment that encourages students to treat the computer like any other design tool. Since studio classes typically meet twelve hours per week, and since the student's desk is his or her primary turf for the term, each student gets far more intensive exposure than they would from machines in shared labs.

THREE TRIAL APPROACHES

To conduct this electronic studio, we combined two classes, the studio course in Architectural Design and the survey course in Computer Applications. Traditionally, design exercises are assigned without much direction to the students about which media they must use. Conversely, computer drawing exercises in applications courses are usually assigned without much direction to students about what content they must include. With these two classes combined, it is possible to devise exercises in which specific strengths of the software are keyed to specific needs in the design process. For example, designers frequently begin their problem analysis by constructing graphic matrices that illustrate the possible permutations of several design options. This can be done with the computer's ability to copy and paste, so that various combinations of physical configurations can be easily generated.

Another example of the benefits of collaboration between these two classes is in the use of 3-dimensional image generation. Most 3-D programs for microcomputers do not accommodate detailed design development. Yet they are very good at calculating crude line drawings in accurate perspective. A powerful use of mixed media is for the computer to generate the time-consuming perspective, while the designer traces over with paper and pencil or in a less sophisticated "paint" program, adding the detail that brings a drawing to life.

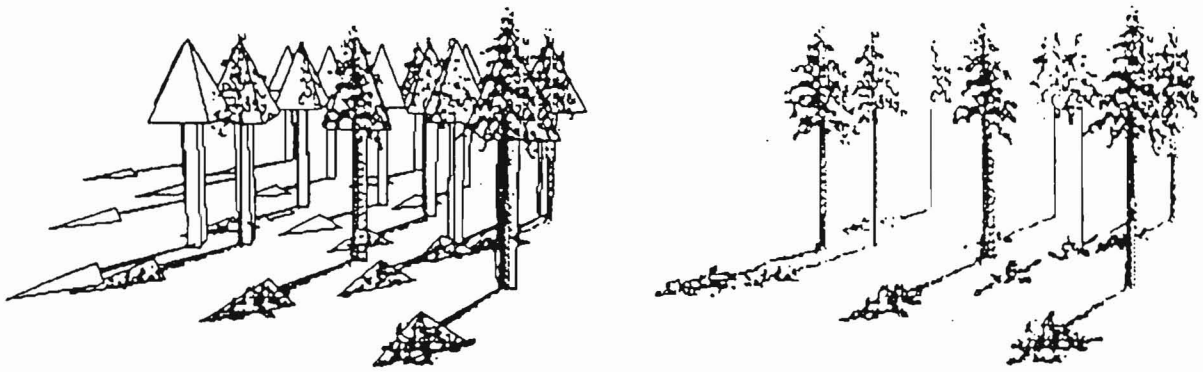

Figure 1. Schema is used to draw the perspective with geometric shapes which are painted in with Superpaint. Then the Schema image is removed leaving the Superpaint image.

By encouraging this kind of media mixing, we can demonstrate the integral nature of computers in design and use relatively simple software in the process.

The design studio was offered at the third/fourth year level in the curriculum. Within this population, there was not a sufficiently large pool of students who had had prior computer experience, so our classes included a large proportion of "computer illiterates". We soon discovered that students learn the basics so fast, with this 24-hour access, that after the first few weeks there was little difference in literacy between those who had had a prior introductory lecture course and those who had not.

Over the course of three terms we tried three methods for introducing the computer to novice students: 1) through an accompanying lecture class, 2) in an integrated seminar paced to the studio, and 3) and with an intensive workshop for the first three weeks of the studio. We tried about twenty pieces of software for sketching, drafting, perspectives, analysis, and prototyping, in combination with basic exercises, drawing exercises, and design exercises.

Because we had taught design and computer applications separately for a number of years, our first experiment involved continuing similar practices. Studio problems were given out in a normal schedule, with flexible media restrictions. Meanwhile, the students took the same applications survey course that was normally offered to the architecture student population at large. To our surprise, though the courses had been enthusiastically received separately, when perceived as a pair, they were not as successful. The students felt overwhelmed by the quantity of material they were given in the survey course, and tended to ignore everything that they couldn't use directly in design. At the same time, few of them learned the software thoroughly enough to be productive, and many of them returned to their drawing boards for conceptual design work. We concluded that the sequence of presenting material in a lecture (going from simple to complex, spaced throughout the term) was inappropriate for studio users, who often needed the most complex software first and who needed it all as early as possible in the term.

In the second term we varied the pattern of the lecture course. We offered an applications seminar that was designed exclusively for the studio students. It emphasized 2-D and 3-D drawing early in the term, with software peripheral to preliminary design (such as word processing and technical analysis) towards the end. We also taught less software. This second paradigm worked better than the first, but students still complained about too much software and not enough concentration on a thorough mastery of the key programs.

Finally, we concluded from the first two approaches that in a computer studio there are three fundamental kinds of learning going on. Not only must the students a) learn the technical aspects of using a computer and b) learn to design, they must also c) learn to *design with a computer*,

which is qualitatively different from either (a) or (b). As instructors, we discovered that this third kind of learning requires far more care and attention than is conventionally given to merely learning the technicalities about using computers. We had begun by thinking that if you satisfy learning needs (a) and (b), that (c) would follow naturally. We concluded that learning to design with a computer was not natural and would require carefully designed exercises. Even though we instructors may be adept at teaching ourselves new software frequently and easily, most students do not have that facility, and teaching must be strenuous and explicit.

AN INTEGRATED APPROACH

In the third term we addressed this revelation by designing a series of specific exercises to encourage students to exploit the unique graphic qualities of the Macintosh while simultaneously developing the ideas and drawing skills needed in the preliminary stages of design. We decided to strip our software library down to the bare essentials. We chose a "paint" program for sketching and diagramming, a 3-D modeling program for generating perspectives, and a 2-D drawing program for final plans and elevations. For each of these three programs we conducted a week-long intensive workshop. Each week was scheduled as follows:

Days 1-2: A lecture/demonstration of the software, followed by intensive, hands-on tutoring and basic exercises to get acquainted with the "nuts and bolts" of the program

Days 3-4: A non-architectural drawing exercise to develop graphic facility

Days 5-7: A design exercise for a very simple building using specified media and rigid formatting requirements. Students were encouraged to concentrate on areas where the computer outperforms what a human draftsperson could do, since this is where studio computers will ultimately be the most useful.

After taking each of the three programs through this intensive, week-long cycle, the students had mastered the media and were ready to spend the remainder of the term on a longer design problem of the complexity that is normally expected at the third/fourth year in our curriculum.

EXAMPLES OF EXERCISES

For each piece of software we created three types of exercises: basic, drawing and design. The basic exercises were intended to help the students experiment with combinations and sequences of tools that are typically used in architectural applications. The tools were used to create abstract images rather than architectural images, so the students could concentrate on learning the "nuts and bolts." The drawing exercise built on the tools and sequences of tools learned in the basic exercises but concentrated on architectural rather than abstract images. The students were asked to draw buildings but not to design them. This allowed them to concentrate on computer drawing techniques without the anxiety of having to create a DESIGN. Finally, the design exercises were short, simple problems that were structured to make the drawing tools attractive and simple to use. What follows is a description of several exercises that illustrate these points.

For the basic exercise, the students designed a "Christmas ornament", a six-sided fold-up cube, 1-7/8" on a side. Each of the six sides was required to contain a simple design that exercised typical procedures that the students were likely to encounter later in their architectural drawings.

Exercise drawn	Lesson Learned
A circle which fits perfectly in a square	Graphic primitives and geometrical relationships
A library of line weights	Object Attributes

A radial design with twelve spokes	Duplication, rotation
A composition of three or more arches	Manipulating graphic primitives to build up more complex composite drawings
Contour lines that touch three sides	Controlling "Freehand" Drawing in an object-based environment
A bitmapped image from a paint program	Understanding the relationship between different Macintosh programs, and how to share graphic information between them.

In addition to these drawing skills, the students also learned to print on colored cover-weight paper with colored printer ribbons, techniques that they later made useful by folding printouts into architectural models.

One of the drawing exercises to teach architectural skills in a paint program was the creation of a building plan and site plan using no lines or outlines, only fill patterns. Because most students have developed a fair facility with pen or pencil by their third or fourth year, and because lines are faster to draw than textures using traditional methods, most preliminary design work tends to be described with line only, totally neglecting the surface character of buildings. In paint programs, by contrast, patterns are easy to draw with, and personal, idiosyncratic patterns are easy to design. As a result, this exercise awakened students to the potential of describing the *surface* of their work rather than the edges, and their drawings were more interesting texturally and provided a more provocative stimulus to their visual thinking. Because the patterns provided in most commercial Macintosh software are unsuitable for architectural drawing, students were strongly encouraged to develop their own, usually asymmetrical patterns.

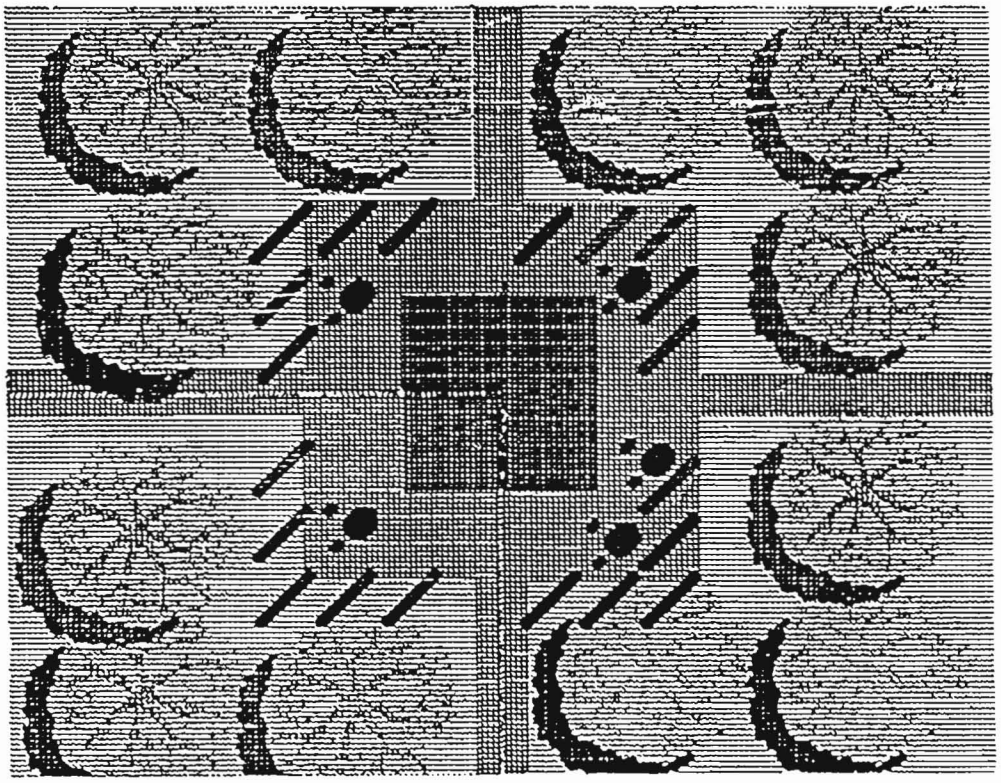

Figure 2. Site plan using patterns.

This exercise also required students to print out their site plan in the middle of four 8-1/2"x11" pages so they could experiment with the program's capability to display several drawing pages at a time.

The drawing exercise for an object-based drawing program was designed to exploit its ability to change scales and print on multiple sheets. Students were asked to create a floor tile at double full scale, make a composite of four-by-four field tiles surrounded by a border tile, print them full scale, and, finally, draw a floor plan of a room at 1/2"=1'-0" using the tiles they had designed. The full sized printouts were placed on the floor in the corner of the room to help the designers imagine what the room they had designed might be like. The jumps between scales with or without preserving pattern detail was an important lesson about the relationships between scales and about methods for manipulating perceived sizes of architectural spaces.

The design exercises for all three pieces of software was a three-day sketch problem to design a "Three Room House of Thermal Delight". The design criteria were kept extremely simple so that students could experience a small, focused success on their initial trial of design with a computer. The presentation requirements were selected to reinforce the basic and drawing exercises. For example, in the paint program's design exercise, the site plans were to be drawn using only patterns.

At the end of the three week workshop, the students had more/less mastered the three principal drawing programs. For the remainder of the term they proceeded with a conventional design problem, in a conventional design sequence. Unlike in the first two terms of this experiment, this third group of students was completely successful in integrating computers into their design processes, from initial ideas to final presentations. A few made brief use of tracing paper, but not one student reverted to conventional parallel rule drawings.

RESULTS OF STUDENT SURVEYS

At the end of each term, we asked the students to evaluate their experience in the Mac Studio. The responses of the participating student designers generally matched the conclusions that we instructors came to on the basis of our observations of their ease of working with the computers the quality of their design work. The main findings were that the building design was little affected by the computer, that the design *process* was greatly influenced, and that the intensive software training in specific computer drawing skills improved the quality of the influence.

Where the design itself was affected by the medium, it was because curvilinear and irregularly-shaped buildings were (or were perceived to be) more difficult to draw than rectilinear ones.

Where the design process was affected by the medium, it was primarily in two areas, *reiteration* and *visualization*. First, the students who mastered the software were able to try more variations than they normally would because each variation took less time. As traditional designers know, the more trials, the better the result. Second, the ability to generate quick perspective sketches enhanced the students' ability to understand the spatial implications of their drawings.

Nevertheless, the most common complaint was that the software was not suitable for the early, loose sketching that characterizes preliminary design, a place in the design process where students spend a large proportion of their time. Most of this work was still done on paper.

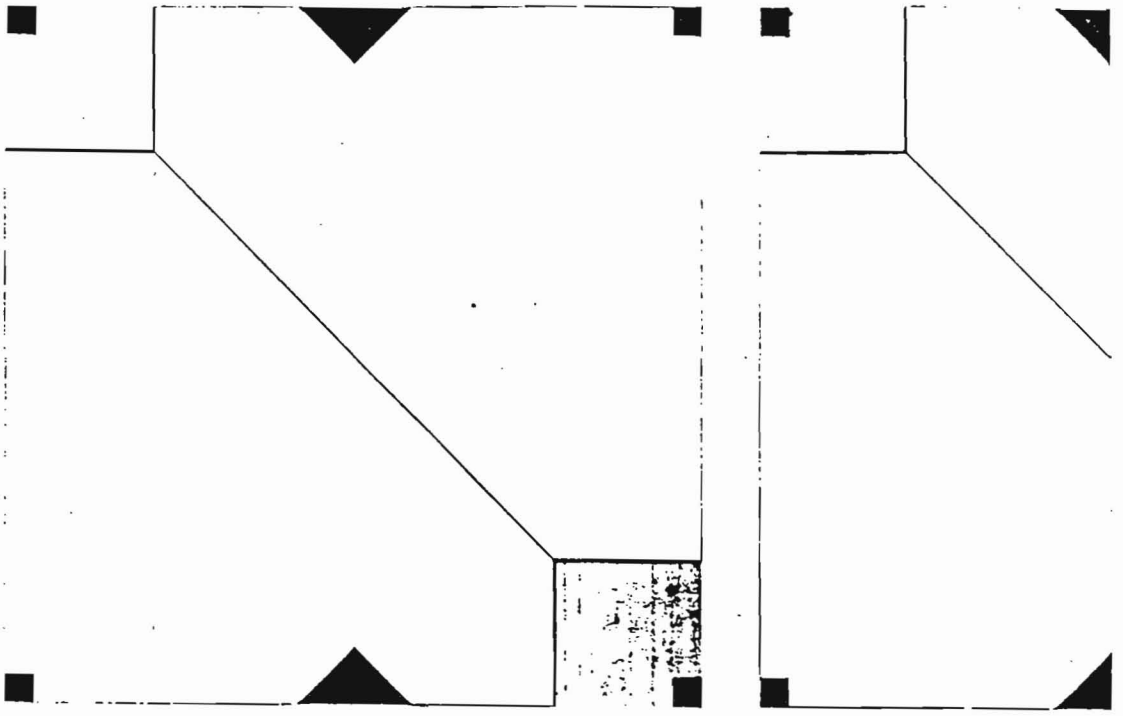

Figure 3. Drawing of full size field file and border tile.

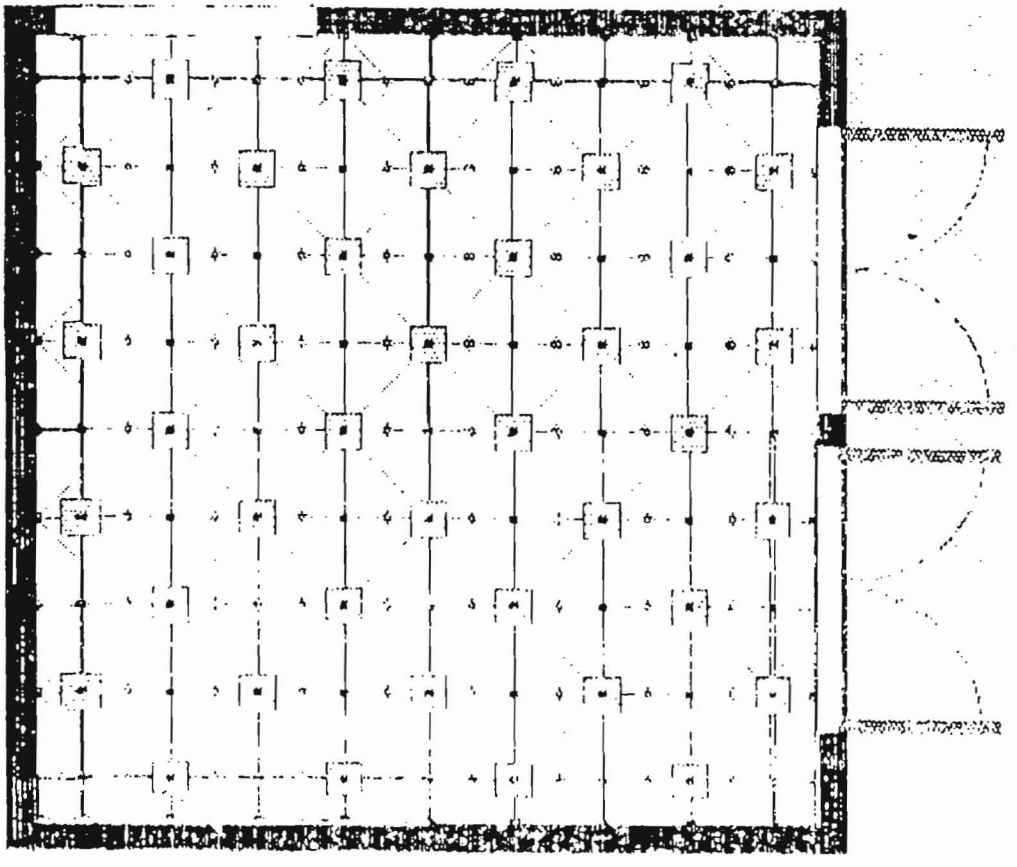

Figure 4. Room plan showing tiles.

CONCLUSION

We are compiling the results of this year-long effort into a studio curriculum based on existing software and available hardware. Because existing software has major omissions and isn't carefully matched to architectural uses, a large part of our work is in the creation of the connecting tissue that relates this software to the studio. The connecting tissue consists primarily of written material: problem statements, design exercises, methods of instruction, software reviews, lecture materials, and evaluations of the fit between software and individual student's learning styles. We are developing design problem statements that best exploit the characteristics of various pieces of software without sacrificing the richness of the learning experience normally encountered in the studio.

This work is important because the design studio is the core of architectural professional education, and changes to its character as the result of powerful new design tools can affect the quality of our built environment. Until now, the computer's primary use in architecture has been for analysis. So as computers come into the design studio, they bring with them a technical aura that biases the studio towards analysis rather than synthesis, which most educators agree should be its primary focus. However, the computer holds the potential to greatly enhance the designer's synthetic abilities if it is exploited within the traditional domain of design studios.

APPENDIX: THE STUDIO ENVIRONMENT

HARDWARE

Our studio configuration was modest but adequate. Each student had a Macintosh SE on this or her desk, and groups of three or four shared a dot matrix printer. Most of the SE's were two-disk systems; a few had 20 megabyte hard drives, a size that was perfectly acceptable for the amount of software used and the level of material the students were able to generate in one term. The printers were connected by simple ABCD switches because it seemed unnecessary to share data and because we had observed problems with other local networks. The studio also had a pen plotter and a laser printer, each with a dedicated SE that doubled as a demonstration machine.

With a more generous hardware budget, we would have requested a 20 megabyte hard drive, memory upgrades, and a larger monitor for every workstation, plus shared peripheral equipment such as digitizing tablets, video digitizers, plotters, and laser printers.

SOFTWARE

We received generous donations from software developers throughout the year and evaluated them in the studio context. We discovered that most of the programs were not useab le in preliminary design. (The detailed evaluations of these programs are beyond the scope of this paper.) Whereas a survey course may cover a variety of graphics and analysis programs, word processors, spreadsheets, and so on, much of this software is inappropriate in studio. For example, using a spreadsheet for cost estimating is very useful in professional practice, but this capability is distracting at best during the preliminary stages of conceptual design.

In our final term of experimentation, we settled on three primary programs for the studio work. We chose SuperPaint, from Silicon Beach Software Inc., as a drawing program because it combines the flexibility of pixel-based drawing with the measured accuracy of object-based drawing. This was the program that students used to draw diagrams and preliminary sketches. Our chief complaint about this software was that it did not support drawing sizes larger than 8"x10", but new versions of this program are presumably remedying this.

For three-dimensional modeling, we chose Schema, the program developed at Harvard by

ACADIA member, Mark Van Norman. Students were able to generate far more perspective presentation drawings than they ever had before, and some who became particularly adept were able to literally design in three dimensions. Although the version we used suffered from a non-standard user interface, it was the only 3-D program we were able to evaluate that was suitable for architectural designers.

The third program we used was MacDraft, from Innovative Data Design, Inc., one of the oldest 2-D drawing programs for the Macintosh, and, in our opinion, the one with the right balance of power and ease of use for preliminary design. In MacDraft, students were able to do measured drawings for their final presentations and, most importantly, import and compose the drawings made in the other two programs.

Students also learned to use MacWrite, PageMaker, FrameMac, and Excel, but these were used primarily for non-studio projects.

WORKSTATION DESIGN

Because our goal was to literally integrate computers into the studio setting, we gave some attention to the question of furniture and ergonomics. Throughout the year, students experimented with a variety of configurations. The variation was as broad as the working styles of the individuals. Some chose ell-shaped credenza relationships between drawing table and computer table. Some chose low tables with chairs; others chose high tables with stools. Some abandoned their drawing boards altogether, and simply used their desks to spread out printouts on. We designed and built several above-desk pull-out shelves for the computers that proved very popular. This allowed the computer screen to be viewed at eye level when in use, but to be pushed conveniently back when the full surface of the drawing board was needed for conventional drawing activities. We had anticipated that it would be important for students to take control over this aspect of their environment, but we seldom saw workstation layouts change once they were settled at the beginning of the term, even if the students grew unhappy with them.

SECURITY

More than in other academic departments, the security of these highly-coveted machines is a complicated issue in architecture, where free communication between students and 24-hour access to studios is considered essential to the design education experience. Unfortunately, we had to isolate this studio somewhat from the rest of the department, which tends to spread out into open-plan studios with few physical barriers in between. Not only was the Mac Studio the only studio on campus with a lock on the door, it was the only studio on campus with a door. We used a lock with a combination that could be changed every term. Students were encouraged to keep it closed at night or when it was sparsely occupied, but to leave it open during the daytime to encourage the normal traffic of friends and colleagues. We also cabled and locked major pieces of equipment to heavy furniture. In the first year of operation, we have had no thefts and one minor incident of vandalism.

The safety of the individual machines was ensured by a few simple rules, which we never found violated. Giving students a sense of responsibility for "their own" equipment was the best safeguard. For example, food and drink were allowed in the room, so as to encourage a normal studio atmosphere; however students were encouraged to keep spillable liquids "behind and below" the equipment. This perceived leniency encouraged students to comply. They were further encouraged to respect equipment safety by the promise that they would need to do the remainder of the term's drawings in one of the university's centralized labs if they damaged their machine. We further helped students to feel "ownership" by disallowing them to share their machine with friends or even spouses. Although there may have been some late-night violations of this rule, there was no evidence of students being "locked out" of their own studio desks

because of their generosity towards friends.

BIBLIOGRAPHY

Brown, G.Z., and B. Novitski, "Nurturing Design Intuition in Energy Software," *ACADIA '86 Workshop Proceedings*, Association for Computer Aided Design in Architecture, Houston, Texas, 1986.

Brown, J. L., "Integrating Computers into the Design Studio: A Critical Evaluation", *ACADIA '87 Workshop Proceedings*, Association for Computer Aided Design in Architecture, Raleigh, North Carolina, 1987.

Edwards, B. *Drawing on the Right Side of the Brain*, J.P. Tacher, Inc., Los Angeles, 1979.

Flemming, U. and G. Schmitt, "The Computer in the Design Studio: Ideas and Exercises that Go Beyond Automated Drafting", *ACADIA '86 Workshop Proceedings*, Association for Computer Aided Design in Architecture, Houston, Texas, 1986.

Goldman, G. and S. Zdepski, "Form, Color and Movement", *ACADIA '87 Workshop Proceedings*, Association for Computer Aided Design in Architecture, Raleigh, North Carolina, 1987.

Harfmann, A.C., L.M. Swerdloff, and Y.E. Kalay, "The Terminal Crit", *ACADIA '86 Workshop Proceedings*, Association for Computer Aided Design in Architecture, Houston, Texas, 1986.

Herbert, D., "Study Drawings in Architectural Design: Applications for CAD Systems", *ACADIA '87 Workshop Proceedings*, Association for Computer Aided Design in Architecture, Raleigh, North Carolina, 1987.

Miller, F., "Architectural Design and Solid Modeling", *Academic Computing*, December 1987/January 1988.

Negroponte, N., *Soft Architecture Machines*, The MIT Press, Cambridge, Mass., 1975.

ACKNOWLEDGEMENTS

This project has been supported by the University of Oregon Department of Architecture and Apple Computer Inc. Additional hardware and software was donated by Zericon, Mark Van Norman, Innovative Data Design, Inc., Silicon Beach Software Inc., Working Software Inc., Aldus Corporation, Microsoft Corporation, Odesta Corporation, SmetherBarnes, J. J. Jordan, Diehl Graphsoft, Inc., and Erez Anzel Software. Illustrations of student work were provided by Deborah Rink-McGinn, and Wei-Zen Hao.