Micro-computers and Computer Aided Design Instruction

Robert E. Johnson
Architecture and Planning Research Laboratory
The University of Michigan

Abstract

During the past few years we have been involved in a variety of experiments that teach the use of computers to non-computer oriented architecture students. These teaching experiences have led to the development of an experimental, entry-level course in computer-aided architectural design using the Macintosh personal computer. Objectives of this course included: a) to provide an introductory course to students with little or no prior computer experience, b) to use the course as a vehicle for illustrating principles of computer-aided design, c) to teach the course so that it would be applicable to design in general, not just architectural design, and d) whenever possible, to use "off the shelf", generic, readily accessible software. The history of these developments will be presented along with a very preliminary evaluation of results.

Some Basic Principles

A large and growing number of students are interested in learning how to use computers for design. They are not necessarily interested in learning computer programming, but are interested in understanding how computers can be used in their work. Early instructional experiments suggested some basic, fundamental requirements for introducing computing to this group of students.

To Minimize "How to" Teaching, Select An Appropriate User Interface

If the user interface feels foreign to previous student experiences, most of an instructor's time will be spent in explaining and demonstrating "how to use it" rather than solving class problems. One of the first, early experiments involved the use of in-house, special purpose software on the university's time sharing mainframe facility as a part of an introductory course on building economics. Most of this software was relatively simple in nature (e.g., data input for one of the early homework assignments). However, we often found ourselves repeatedly explaining some of the most fundamental aspects of computing (for example, the logon procedure). Teaching the use of computers in this type of situation can begin to resemble a course in driver education.

Some authors (Kay, 1984) place a great deal of emphasis on the user interface, and it does seem that there are some fundamental user differences that can place artificial barriers in the way of learning about computing. One of those barriers is that some (perhaps most) design students have a relatively low level of keyboard skills. Learning the syntax and finding the right keys for a text-driven command parser can represent a significant problem for these students. On the other hand, secretaries, who frequently have a high level of keyboard skills, generally ignore the availability of pointing devices and prefer to interact with machines using keystrokes.

In the Winter term, 1985, the computing center installed a cluster of 15 Macintosh personal computers in the Art and Architecture Library. Since these would be extremely convenient for student use, we made arrangements to purchase 10 copies of Multiplan through the College. In many ways the "spreadsheet paradigm" was ideal for a course in building economics. Many of the problems are conceptualized in a table format and require relatively simple arithmetic calculations. However, as we began to introduce students to this computing environment, we noticed more questions relating to the problems to be solved and fewer questions about how to use the hardware and software. It was concluded that a visual interface was more appropriate to design students and "first time" users.

To Maximize Leverage, Teach Generic Tools

It is not possible to completely eliminate all the "how to" questions and answers. However, if software is specific to a particular application or problem domain, most knowledge gained about the use of the tool will quickly atrophy after the class has ended. "Use it or lose it" applies in this case. This probably contributes to the explanation as to why some students could not
remember the procedure for logging on the main frame computer. They simply did not have any reason to do so on a regular basis, and therefore forgot.

During the Fall term, 1985, the use of Multiplan was introduced in a more organized fashion into the course. Each Friday became a laboratory session where the class could discuss specifics of the problems assigned as well as the use of Multiplan and the Macintosh in solving those problems. During this term many students began to use their spreadsheet knowledge in other courses — and even in solving problems outside school. To encourage this behavior Multiplan problems outside the subject matter of the course were occasionally discussed. These experiences suggested that there may be a great deal of leverage in teaching students how to use "generic" as opposed to special purpose software.

Development of Micros in CAD Course

In a separate development, I had been asked during the summer of 1983 to guest lecture for several sessions in a course with the title "Microcomputers in Urban Planning". This course was intended to be an introduction to the IBM PC and standard PC software such as WORDstar, Lotus and dBASE III. My role in this course was to introduce students to equivalent software on the Macintosh. By the conclusion of this course I felt that there might be enough interest in the part of students to offer a similar course, but focused on the principles of computer-aided design.

It was also during this period of time that I became aware of a course taught in Mechanical Engineering at Stanford by David Thornburg with the title Computational Tools for Creative Designers. The objective of the course was described as covering "... the topic of computer-based tools from the perspective of those whose thinking style is largely non-analytical in nature." The course was taught using ExperLogo, and attracted students from a variety of backgrounds. It was a programming course with an objective of teaching three basic topics: 1) The representation of knowledge in a machine, 2) The philosophical aspects of program design and 3) The practical aspects of creating and debugging programs. While the course resulted in some interesting projects, it was acknowledged that most of the time was spent in the practical aspects of creating and debugging programs.

A final factor that influenced the design of this course was a series of discussions and reports that had been occurring within the university about the general issue of needed changes in the computing environment. One of the major recommendations was to develop computer instruction for greatly increased numbers of students (LSA Report, June 1, 1984). It was acknowledged that there are levels of training that would be required to attain this objective.

Level 1: Instruction for computer science concentrators.
Level 2: Serious programming training for non-concentrators.
Level 3: Courses that emphasize microcomputer applications for end users.
Level 4: "Short-courses" that emphasize a single application, such as text editing.

Level 3 seemed to be particularly relevant, since it attempted to meet an apparent demand by students for an introduction that is broader in scope and less programming intensive than Level 2. Such a course is currently being taught in the College of Literature, Science and the Arts and is attracting a large number of students (Galler, 1986). Software introduced in this course includes MacWrite, Verbatim, MacPaint, MacDraw, Terrapin Logo and Excel. The text used in this course is "Computers and Information" by Lawrence S. Orilla.

Based on these prior experiences, the initial inclination was to use standard, off-the-shelf "generic" software as a means of introducing students to concepts of computer-aided design. It was decided to use the Macintosh because of its accessibility to students and its apparent ease of use for the beginning student. However, it is quite clear that a parallel course could be taught using IBM or IBM compatible equipment.

Course Organization

The experimental course was taught in the summer term, 1986. Due to a series of administrative problems the course was not officially listed in the university time schedule. As a result, students learned of the course primarily through word of mouth and were required to obtain written permission to sign up — as if they were taking a personal tutorial from me. Despite these obstacles, twenty-four students enrolled in the course. Slightly more
than half of these students were from architecture, and the remainder were from a variety of other university departments, including landscape architecture, urban planning, civil engineering, and computer science. Some of the students had virtually no prior experience with computers, while others viewed themselves as having had a great deal of prior experience. The course was divided into three major sections: design drawing/drafting systems; design evaluation systems; and design database systems.

The focus of the course was to develop a sense of some of the issues involved in computer-aided design by using existing, generic software. Each software package was explored through a combination of lectures, demonstrations and an instructional laboratory. The lectures and demonstrations were facilitated by the use of a projector system that allowed the screen image to be viewed by the entire class. The instructional laboratory involved a series of short exercises intended to familiarize the student with the mechanics of the programs.

Experimentation with the software occurred within the context of attempting to execute the design of a small building — a highway tourist rest station. A small scale design problem was selected so that it would be easy to understand by all students, regardless of their lack of expertise in architecture and construction. The laboratory problems roughly corresponded to stages in the design process:

**Program and preliminary sketches.** Use of MacWrite to develop a program of requirements and initial conceptual sketches using MacPaint.

**Conceptual design.** Use of MacSpace in developing three dimensional concept designs.

**Detailed design.** Use of MacDraft in developing plans, elevations and details about their designs.

**Design evaluation.** Use of Excel to “test” their design by developing various evaluation schemes.

**Design evaluation and design databases.** Use of Helix to develop a more comprehensive database approach to design evaluation and to discuss concepts of design databases.

---

**A Preliminary Assessment of the Course**

This section discusses each laboratory exercise in greater detail, together with an evaluation and assessment of each section. Because of the small number of students involved in this experiment, the evaluation is largely in terms of anecdotes and impressions rather than statistical inferences. Selected samples of student work are included to give you a flavor of the type of work that was produced in the course.

**Lab #1: Program and preliminary sketches**

This initial section of the course was intended to be a “warm up”. It was important to start slowly, since some students had virtually no prior experience with computers. Furthermore, this section gave all students a basic familiarity with MacWrite, which was to be the major tool used for preparing laboratory reports. During this section of the course we discussed some of the basic hardware requirements of CAD systems and where micros fit within this; how people interact with computers for drawing and some of the advantages and disadvantages of different interaction methods; as well as an introduction to the use of the Macintosh and the software.

![Figure 1](image)

*Figure 1: MacPaint sketches*
MacPaint allowed us to introduce the concept of the metaphor (Kay, 1984) and to understand how it was possible to provide the illusion of paper and pencil within a computer program. We began to discuss the meaning of integrated computer-aided design as we used the Switcher to cut and paste our drawings from MacPaint to MacWrite. There was a great deal of enthusiasm at the start of the course. As a result, the simple conceptual sketches that were expected turned into fairly elaborate and detailed final designs.

Student Comments:

• MacPaint is limited in that it is hard to change a drawing once it has been created.

• Some mistakes (which may happen to anyone) interrupted the creation of an idea. Paper and pencil do not have this problem.

Lab #2: Conceptual design

Prior to the introduction of MacSpace, we discussed various approaches to the design of physical objects - none of which are mutually exclusive. Some of these included:

• Combining "primitive" objects to develop a more complicated object.

• A focus on spaces (voids) and the layout of those spaces as opposed to concentrating on the planes and surfaces that enclose those spaces.

• The arrangement of planes and surfaces as the primary focus of design. (The opposite of the space layout approach.)

• A sculpting approach to design, where the designer carves out a design much the same way a sculptor works.

• A modular or grid approach, where a grid of some kind is used as an organizing device. Functional modules or structural grids would fall into this category.

The purpose of this discussion was to explore the multiple methods and sometimes idiosyncratic approaches that are an inherent part of the design process - and how the computer may or may not address these issues.

As we explored the use of MacSpace, we discussed the issue of how to accurately input a...
three-dimensional object using essentially two dimensional tools. This discussion centered on standard techniques such as extrusion, revolving surfaces around horizontal and vertical axes, and the necessity, at times, of typing coordinates. We discussed the different approaches to three-dimensional modeling and compared the differences between the Boundary Representation and Constructive Solid Geometry approaches. Easy3D was used to illustrate the concepts associated with sculpting objects (union, intersection and difference operations).

Student Comments:

- Computer programs, in general, prefer straight lines - they don't like free hand sketches and curved lines (like domes).
- The program was frustrating to learn at first. I spent six hours working with the program on Friday and left somewhat discouraged. But it seems that getting away from the computer for a while helps one gain a new perspective on it.
- The Macintosh screen in general is too small for many of the graphic functions.
- It didn't feel user-friendly.
- If you have to type in each coordinate and be very specific about your design, that means you have already designed your building.
- I spent more time drawing it on the computer than I would have if I had drawn it by hand several times.

Lab #3: Detailed Design

The discussion for this section of the course centered largely around the principles of organizing two-dimensional drafting systems, including: Basic drawing tools (point, line, polygon, circles and ellipses, symbol libraries); Annotation/modifications to base drawing (text, dimensioning systems, polygon fill, fillet and chamfer); Basic transformations (translate, rotate, scale, mirror, instantiation) and input techniques (rubber-banding, typing coordinates, snapping, grids, layers, zoom and pan, viewports, clipping). Students did most of their work using the capabilities of MacDraft. However, they were also introduced to more complicated systems, such as EZ-Draft and AutoCAD. AutoCAD running on an AT clone was available on an optional basis for one week during the term.
Lab #4: Design evaluation

This was a particularly critical part of the course. Prior to this section, almost all the focus had centered on drawing systems of one kind or another. The enthusiasm for these early parts of the course was obvious from the richness and variety of design solutions displayed on the walls of the classroom. However, the philosophy of the course was one of comprehensive design, requiring an evaluation and testing of proposed design solutions (Simon, 1982). Part of this evaluation process is visual and qualitative, and part is quantitative. Slightly complicating matters was the fact that not enough attention had been paid in the original project description to specific design criteria and there had not been enough time to think through and develop a meaningful set of design evaluation procedures (the instructor's fault - one of the "glitches"). Therefore, when it came time to evaluate the designs, the suggested criteria were not meaningful for all the designs (for example, performing a degree-day heat loss evaluation when your design does not have doors and windows).

Nevertheless, we were able to develop and illustrate some concepts about how evaluation could be integrated into a computer-aided design system. We used as a starting point a discussion of the "traditional" design process - using documents that had been created by a practicing professional in the design development of a house. We discussed the types of design evaluation that drawings are particularly good at supporting, such as locating doors, furniture and sizing spaces. We also discussed design evaluations that traditional design representations (drawings) do not provide much help with, such as cost calculations and energy simulations. Ultimately, we concluded that each method of
developing/representing a design has its own built-in bias. From this starting point we began to discuss the need for multiple representations of designs — a drawing being only one of many possibilities. We began to discuss the concept that the actual design is something other than a drawing. In the conventional sense it is defined legally as a set of construction documents (plans and specifications). In a more comprehensive sense it is all the information and knowledge (spatial and non-spatial) that contribute to the definition of the parts of the building and how they will operate once it is constructed. In short, we began to think of design as the process of constructing a database which is then evaluated as to whether or not it satisfactorily meets the criteria (both stated and unstated) of the decision-maker. We also began to see that this database is not static in nature, but that it changes as one progresses through various stages of design and continues to evolve throughout the life of the building.

We introduced students to all aspects of Excel, including database, chart and macro facilities. Excel was useful in illustrating basic principles of database design as well as the notion of alternative representations of designs. It was also useful in demonstrating the difference between a "black box" evaluation system and a "glass box" approach. As a side issue, we discussed concepts for designing charts and graphs (Tufte, 1983).

Student Comments:

• Excel’s database and macro features can be enjoyable to use once the initial confusion has worn off.

• Its applications are almost limitless, but it does seem to be extremely useful for architectural design evaluation models.

• In my limited (very) knowledge about computer programming in general, all I can say is that I personally found Excel to be very appropriate and useful for what I would need.

• Before I came to the Michigan I was involved in research on the evaluation of a health clinic building. The evaluation took the effort of two people for one month. With the simple steps in this report, I would have been able to do the whole study in less than one week with more accuracy.

• I think it is not possible to know everything about this program unless there is a need for it.

Lab #5: Design evaluation and design databases

Helix was selected as the database for several reasons. First, it is marketed as a "relational" database and exhibits at least some of those characteristics. In keeping with the desire to focus on "generic" tools, a relational database was a primary requirement. Helix also allowed one to have "picture" fields and had an advanced capability to custom-design output forms. I felt that the visual characteristics of Helix would have a greater chance of appealing to designers. Finally, I was not convinced that the use of icons within Helix was particularly successful, and I was interested to see how students would feel about this.

Most of the class discussion centered around the principles of database design with examples relating to design problems. Design data was discussed as being either project dependent (for example, the location of a wall within a particular design) or project independent (for example, the "palette" of available wall materials that exist independently of any particular project). We discussed the possibility of design as a choice process, and how databases can help facilitate the selection of elements based on predetermined criteria. The remainder of the class discussion centered on the use of Helix, including the design of forms and the uses of various icons and calculation tiles. During the class sessions, Helix was perceived as being a more complicated program than the others.

Student Comments:

• I found the Helix program easier that I had anticipated and a bit of fun.

• The time devoted to this software is too small in comparison to its potential capabilities.

• As I worked through the lab, I found Helix to be easier and more interesting than I expected. To be able to include pictures as part of the database was one of its best features.

• The use of icons is not as intuitive for Helix as it is in other programs, but it proves adequate with some practice.
Strengths and Weaknesses of the Course

We were able to introduce students to a wide variety of software and design principles through this course. As with any new course, there were the inevitable "glitches" that are obvious candidates for improvement. Some of these include:

Improvements in instructional laboratories and lectures. The approach of the course is almost one of a design studio, with most learning taking place by doing. The separation of lectures (with projector) and instructional labs represented a long cycle between demonstration and hands-on experience. An improved arrangement would be a closer integration of the instructional lab with the lectures. Following a short introduction, a series of small, short term exercises would be given after which there would be a comparison of alternative approaches and a discussion of the appropriateness of the tool to design.

Laboratory reports. Lab reports were an important means for having the student think about some of the critical design process issues raised by the software. However, the labs need to be designed so that there is less emphasis on writing, and more on doing.

Improved design of laboratory exercises. The course was developed as I went along. Because of this there was not really sufficient time to develop the laboratories as much as I would have liked, and to some extent the laboratory exercises lacked continuity and meaningfulness.

More flexibility in selection of design projects. Since a large number of students were not from architecture, more thought should be given to the potential for students developing their own design projects. It became evident during the term that more discipline-specific design problems were essential if the course were to interest a broad spectrum of students. A possible problem with this is the difficulty of developing an appropriate project independent database for the last laboratory project.

More time. The course introduced a lot of software to students over a fairly brief period of time (two months). It became a very time-intensive course. It may not be possible to do much about this, but one should investigate possibilities for reducing the work load. One way of achieving this would be through the redesign of laboratories.

Overall evaluation. In general, students were supportive of the objectives of the course and as a whole reported to have a good learning experience (see Figure 6). Students seemed most intrigued with the graphics software (Figure 7), although that may have been partly a function of the need to improve the labs in the latter sections of the course. Despite the obvious limitations of using small systems (hardware and software), I found that it was possible to illustrate most principles of computer-aided design. The advantages of small systems (less costly and therefore more readily available, perhaps less intimidating) may make it reasonable for entry-level students. Although "generic" software does not always do exactly what you would like, the obvious trade-off is that you can "reuse" your skills in a wider variety of problem settings. Finally, although some of the students were obviously first-time users, they were able to develop a basic understanding of a wide range of tools and CAD concepts within a fairly short period of time.
Acknowledgements

I would like to acknowledge the help of two teaching assistants, Ivo Wenzler and Uk Kim, in the development and teaching of this course. Their ideas and suggestions about how the course should be developed and taught were helpful and appreciated.

Selected Bibliography

Date, C. J. Database: A Primer. Addison-Wesley: Reading, Mass, 1983.


Thornburg, David D. "The Left Hand of AI: Bringing Design Students to Logo", 14 pages, No date.