The User Interface in Programs for Design Education: Issues and Criteria

Mark Van Norman
Laboratory for Computer Graphics and Spatial Analysis
Harvard Graduate School of Design
48 Quincy Street, Cambridge, MA 02138

Abstract

Due to inexpensive mass-marketed microcomputers and CAAD software the type of "clients" we serve as CAAD educators will soon change. In addition to teaching CAAD programming to 20 students a semester, we may soon be serving a much larger group of casual users from design studios and technical courses. These casual users will require that we provide programs and hardware which allow them to design a better product more swiftly and with less effort than by hand. The most crucial factor in meeting these criteria is the quality of the user interface of the programs and equipment we provide.

At Harvard, we have studied the user interfaces of more than 80 programs used in 10 areas of design. This paper is a summary of a 90 page report in which issues are raised, the answers to which determine the quality of the user interface of a program. In the summarized report, different approaches to resolving each issue are discussed, but no "answers" are provided. In our roles as authors, teachers, and now, consumers of CAAD programs, we must - explicitly or by default - address these issues before designing or purchasing programs and hardware for design education.

Changing Roles in CAAD Research and Instruction

The five years since the first ACADIA workshop have witnessed a dramatic change in the role of the CAAD professional working in an academic setting. This change is due to three major factors: changes in the hardware used in CAAD, changes in the software available for CAAD, and changes in the nature of the users of CAAD software in schools of design.

Perhaps most dramatic are the changes in the hardware used in CAAD education. Five years ago, CAAD use in most schools of design was limited to some form of the "computer ghetto" - usually in the basement, sometimes in another building entirely. In this facility, one or two professors were surrounded by a few advanced students, all of them busy writing experimental software for an expensive mainframe or supermini computer. Schools of design not having the grandest budgets (except for externally financed computer labs), the use of these computers was often timeshared over relatively slow (less than 9600 baud) telephone connections, with the school owning only a terminal and hardcopy device. Due to this slow timeshared computer environment, interactive graphic design in the sense that we know it today was impossible.
Today, the hardware environment of the design school computer facility has (or soon will have) changed dramatically. While the computers may still be in a ghetto for insurance reasons, the systems themselves are now entirely different. Rather than the design school supermini costing $300,000 (not counting maintenance), the school can now provide its students with scores of microcomputers for the same price. Each of these micros, while not capable of the same benchmarks as an unloaded supermini, is capable of much better performance than a supermini loaded with twenty to thirty users. In addition, entire studios can use these micros, where only two or three students might have had access to supermini graphics terminals before. Further, the response of a graphic microcomputer is not limited by the baud rate of the connection to a host. Memory costs have plummeted, and with this, microcomputer graphics with resolution acceptable to (although hardly delighting) designers have become possible. In short, the graphic microcomputer has provided the basic hardware needed to get CAAD out of the basement and into the hands of the average design student.

The average design student, however, is not (and does not want to be) a computer programmer. Fortunately, the advent of microcomputers has been accompanied by an explosion in the availability of commercially produced programs aimed at performing tasks for the design professional. Like the hardware environment, the software environment has changed dramatically since 1980. In the profession, the use of computers for graphic design has slowly begun to accompany the "traditional" uses for accounting, word processing, and engineering analysis. A more subtle change in the orientation of professional software has been an increased use of computer graphics as a fluid input medium, as well as for the display of graphic databases.

The means used to distribute commercial software have also changed. Five years ago, most CAAD software was either produced in house by major architecture and engineering firms for their own use on their own hardware, or CAAD software was sold bundled with the mainframe or supermini computers on which it ran. Now, interactive graphic design software costing on the order of $1,000 is available for microsystems costing on the order of $5,000. Together with new, cheap microcomputer systems this cheap, essentially mass-marketed design software allows the designer not only to purchase his or her own hardware, but also to shop around in a highly competitive market for relatively inexpensive design support software which best serves his or her needs.

As a consequence of relatively cheap mass-marketed hardware and reliable, well documented, inexpensive software for design becoming available, I believe that the nature of our "clients" - both students and faculty - will undergo a rapid change. In addition to writing new programs, teaching CAAD programming, and sponsoring advanced student CAAD projects, we are on the verge of serving an entirely new (and much larger) user group: the casual user from the design studio or technical class. In order to serve this large potential new user group, our own roles as researchers and teachers of CAAD must change. We must go from being
The Importance of the User Interface Due to This Change

With this change in the nature of the students and faculty whom we serve, the quality of the user interface of the programs used becomes of paramount importance, regardless of whether we purchase or write the programs. Previously, getting publishable graphic output from our lightly documented "proof of principle" programs by nursing them past mysterious bugs was acceptable research practice. However, the author and principal user of a CAAD program are no longer likely to be the same person. Indeed, we can no longer count on our student and faculty users knowing anything about how and why the computer does what it does.

Faculty and students in studios and technical courses cannot afford to spend the months it may take to become sophisticated users of CAAD programs, so programs which require arcane codes, memorized command sequences, alphanumerics of coordinates, or other time-consuming tasks are not likely to be used by this new group of CAAD users. A user interface which the computer-naive student or faculty member can understand and productively use is the first prerequisite if a program is to be accepted by the casual user.

Given the opportunity presented by the new hardware and software, our new task as CAAD instructors and researchers becomes the integration of CAAD systems into design studios and classes on an everyday basis. As many of you know, carrying out this task requires a good deal of salesmanship as well as a viable product to offer our potential users. The "sales" issues which I have found matter the most to students and faculty at Harvard are:

* Will the results of using this program be of equal or greater graphic quality than those obtainable by hand?
* Will using the computer allow me to produce the results in less time than if I did the job by hand?
* Can I learn the use of these programs fast enough to get decent studio results by the end of the semester?
* Do I have to learn a whole new way of designing in order to use these programs?
* The ever popular "Why can't I use the machines for word processing?"
The answers which we are able to give to these questions are dependent on the quality and ease of use of the user interface of the programs we provide. The student's desire to use the machines for word processing indicates that many word processing programs already meet many of these criteria. The fact that there is a seemingly bottomless demand for word processing provides a major clue that, if we can meet the needs indicated by these questions, the demand for CAAD in studios and classes should blossom.

Issues and Criteria for User Interfaces in Design Education

In this belief, The Harvard Laboratory for Computer Graphics and Spatial Analysis (Laboratory), supported by a grant from the Academic Information Systems Division of International Business Machines, has set out to determine a set of issues, the answers to which determine the quality of the user interface of a program to be used in design education. In our roles both as producers and consumers of design software, we all must implicitly or explicitly address these issues. This report, however, is an attempt to raise and discuss these issues, not to settle them. It is not the intention of the Laboratory to promote any single approach to the design of the user interface. Instead, we want to raise a broad set of practical issues which should be considered in either the design or selection of a user interface for programs used in design education.

The final report of this study is some 90 pages long. The remainder of this paper is a summary of the final report. The final report starts with a discussion of basic issues which must be addressed before beginning to consider issues about the design of the user interface itself. There are four such basic issues:

1) What is a user interface?

For the purposes of our study, the user interface of a program will be broadly interpreted to consist of the hardware and software used to mediate the interactions between a user and the program being used. Hardware will be considered to consist not only of computer equipment, but also the furniture and lighting components of the workstation. These components can be crucial to the user interface of a program, as they can dramatically increase or decrease the physical comfort and convenience of use of that program. The user interface software of a program will be considered to consist not only of the input and output procedures, but also the manuals and any other program documentation. The Laboratory's experience writing, using, and teaching user interfaces indicates that the best user interface hardware and software may well be useless if accompanied by inadequate documentation.
2) Who are our users?

Our users are students of architecture, landscape architecture, and planning. However, the majority of programs produced to date for use on microcomputers in design have been produced by and for design professionals, not by programmers for students. The nature of the user interface in current programs typically reflects this orientation.

Even though microcomputers have opened up the possibility of CAD use to professionals and students of design alike, significant differences between these two user groups exist. The professional user of a drafting system or architectural analysis program, for example, has a technical knowledge of the field which cannot be assumed in the design student. Programs which may be useful to the professional may be useless to the student because the student may be unaware of notational conventions or standard assumptions embodied in the program. Ideally, programs written for students would embody the pedagogical characteristics of "courseware" as well as the characteristics of the professional programs the students would eventually be using.

An additional difference between the student and professional user of a program is the tolerable "learning curve time" for the program. In a firm, the user of a new drafting system may require up to six months of eight hour days to acquire proficiency in the use of the system. In a professional context, this long learning curve may make perfect sense, considering the potential productivity increases over the next few years. The same program may be utterly useless in an academic context simply because neither the faculty nor the students in a school can spare the requisite time from a full curriculum to developing facility with the use of the program.

A more subtle difference between professional and academic users of computers is the intent of the prospective user. Design professionals and vendors of programs sometimes believe that the proper purpose of design educators should be to train their students in the facile use of drafting systems and analysis programs, thus sparing the professional firms the task of training. Many design educators, however, consider their task to be the teaching of the design and analysis fundamentals underlying the use of the tools, rather than teaching students the use of one or another particular tool. Therefore, many schools will not teach students to use a particular drafting system or analysis program until it can be shown that such use would allow the student to perform analyses which would not otherwise be possible or to design in ways which were not previously possible. There are programs coming onto the market which do promise to permit new design and analysis procedures, but these programs are generally aimed at professional users rather than students, and thus may have the pedagogical shortcomings noted above.
Faculty and staff are also critically important new users of the user interfaces addressed in this report. Faculty will not only be faced with learning the user interfaces, they must also become facile enough to build sample databases and demonstrate the use of the programs to their students. Computer support staff will also be faced with the task of mounting the software, maintaining the files and hardware setups required, and answering the questions of bewildered students.

3) What kinds of computing tasks will the users be performing?

Design students will be performing a variety of computing tasks in support of their technical coursework and design studio work. The most complex tasks will involve the construction of databases representing the site context of the students' project, the modification of the databases to include the project itself, and the consequent analyses of the databases using a wide variety of architectural, planning, and landscape architectural analysis programs.

Currently, complex spatial databases are built using drafting systems, three-dimensional solid modeling systems, and (to a lesser degree) two dimensional paint programs. The structural, thermal, site, civil, engineering, cost, and other analyses are carried out using independent analysis programs. A major problem with the effective use of Interactive Graphic Database Management (IGDM) programs together with analysis programs is that most design analysis programs cannot take the databases built by IGDM programs as input. Almost all analysis programs have their own independent sets of procedures dedicated to the specification of the project with respect to the proper variables for a particular analysis. The user of an acoustic analysis program must first build a model of the project specifying those variables needed to determine the acoustic performance of the project. Usually, these model building procedures are part of the acoustic analysis program, and cannot be used by other analysis programs. The user is therefore faced with specifying spatial aspects of the project for each new analysis program. If a large number of analyses are performed, redundant specification of the form and attributes of the project may be necessary.

The most complex and time-consuming tasks undertaken by these students on the computer for the next few years are likely to be the use of IGDM programs to build and modify the spatial databases necessary for predicting the appearance, cost, structural performance, energy performance, acoustic performance, and other technical characteristics of their projects. For this reason, the user interface of the IGDM program will be considered the "prototypical" interface. The user interface for the IGDM program is certainly the most demanding to design, and in addition, is probably the interface most often used by the designer.
4) In what kinds of environments and for what periods of time are students going to be using these user interfaces?

In our experience at the Harvard Graduate School of Design, most students in first professional degree programs in design still do not own their own computers. The GSD does not require that students purchase their own microcomputers, so we must provide computers at the Graduate School of Design itself. Therefore, our students will learn the use of computers in a room filled with 10 or 20 microcomputers and associated peripherals occupied by 10 or 20 other students at a time. If experience at Harvard and Berkeley is any guide, students work on the computers for periods ranging from 1 hour to 6 hours at a time, duration depending largely on the time remaining until the assignment is due. Time is a limitation, as there are not enough computers to give each of the students as much time on the computer as he or she might want. In addition, faculty and staff help is limited, so students cannot count upon the presence of someone who can answer their questions. The physical ergonomics of the workstation itself constitute another limitation, as space for placement of computers and the large-format hardcopy used by design students is in short supply. Also, computers are likely to be placed on existing tables not designed for computers and chairs are likely to be chosen from whatever is at hand. Consequently, the physical computing environment in design schools is likely to be less than optimal.

Study Methods

Despite a relatively rich literature which indicates that the design of the user interface is a hotly pursued topic, the practical design of the user interface, particularly for interactive graphic programs, is still more art than science. Most of the more than 400 documents in the literature surveyed for this report [19] tend to address the design of alphanumeric interfaces such as those used in word processors and computer mail systems. The design of highly interactive graphic/alphanumeric interfaces for commercial IGDM programs requires that attention be paid to many hardware and software issues not addressed in this literature. Indeed, many issues crucial to the pedagogic success of a program in studios and courses, such as the quality of program documentation and the configuration of the design workstation, are not considered part of the user interface at all. Consequently, one major source of the issues raised in our study is my own experience at writing, teaching, and using IGDM and analysis programs serving designers.

Another major source of issues is the comments of faculty members and student research assistants who, as part of the project, mounted, learned, used, and documented their experiences with more than 80 commercially available IGDM programs. Their experiences and evaluations of the pedagogical quality, design utility, and user interface of more than 80 programs serving ten subject areas in design are documented in a series of reports published as the major result of our project. [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18]
Many of the issues raised in this report have yet to be explored in controlled experiments. Hence, positions described for these issues will often be based on the "best professional judgement" of the author. They are not intended to represent the views of Harvard University, the Harvard Graduate School of Design, or of the Laboratory for Computer Graphics and Spatial Analysis.

The issues are arranged in groups according to the following outline:

1) Software Interface Issues
   - Issues Relating to Program Structure
   - Screen Layout, Prompting, and Data Entry Issues
   - Issues Concerning the Relationship Between Program and Operating System
   - Issues Concerning Response to and Recovery From User Errors
   - Issues Concerning the Uniformity of User Interface and Program Structure Across Applications

2) Documentation Issues

3) Hardware Interface Issues
   - Input Hardware and Hardware Control Issues
   - Output and Storage Hardware Interface Issues

4) Workstation Design Issues
   - Issues of the Physical Ergonomics of the Workstation
   - System Configuration and Cabling Issues

The issues addressed in this report are phrased wherever possible as questions about the user interface of any IGDM program or hardware which, we believe, should be considered by anyone writing or purchasing such a program. In some cases, such phrasing is not possible, usually because issues involving more than one program are addressed. Laboratory staff identified more than 50 user interface issues of interest to the designer or purchaser of IGDM programs for design education. Due to space limitations in this summary paper, only the headings of the issue groups are shown above. To give the reader the flavor of the way in which the issues themselves are treated, two sample issues are presented here.

162
In programs requiring graphic input of existing data on paper and those which output data in the same fashion, the user often needs a good idea of the "scale" of the graphic database in the context of the final output of the program. How is this sense of scale provided?

Discussion
When our research assistants were questioned during and after use of ICDM programs, they reported confusion as to the scale of the "drawing" on which they were working. The confusion was due to the common (and necessary) "zoom" and "pan" functions used in these programs. In addition, confusion was sometimes generated by the superimposition of an "architect's scale" or "engineer's scale" over the screen. Another source of confusion was the capability offered by some programs of scaling plotted output further when actually plotting. Still more confusion was generated when program documentation undertook an explanation of the internal data units used by the program. The issue posed here is that of providing the user with a relatively simple method of specifying the scale and size of the final output.

Position 1
The user should specify the output size of the drawing in real inches at the inception of the database creation. Subsequent zooms and pans should not affect this originally specified size. All areas and distances should be given in inches or centimeters as they would appear on the final hardcopy. This position has the advantage of relative simplicity, given the fact that the user need only bear a single "scale" in mind. The disadvantage of this approach is that standard architectural scales cannot be used when autodimensioning is a feature offered by the program.

Position 2
The user should specify the output size of the drawing in real inches, together with drawing scale, at the outset of the drawing. Subsequent zooms and pans will not affect the scale of the drawing, and distances and areas will be computed and displayed at the specified scale, regardless of the current zoom setting. For example, if a drawing scale of 1/8" = 1'-0" is specified, a line one inch long on the output would be autodimensioned at 8'-0", regardless of the zoom applied to the view of the drawing when the autodimension was generated. This approach avoids the problems of autodimensioning inherent in the last approach.
Position 3

No "scales" are offered at all. All input is in arbitrary system units. This approach has the advantage of eliminating time consuming dimension conversion and display routines and may, in fact, be adequate for 2 and 3 dimensional "sketch" programs such as paint programs. In these programs, exact scaling is not as important as the spatial relationships among elements of the database. This approach might work well for "what you see is what you get" displays, where the eventual appearance of hardcopy is a simple, direct scale transformation (at most) of the appearance of the screen.

Position 4

The user specifies the dimensions in world units of the boundaries of the database, whether these dimensions be in feet, miles, or microns. The program converts these units to internal units in such a way that the resolution of the system is maximized. For example, the user might specify that the database is to be 200 feet by 200 feet. A 16 bit integer system can represent about 64,000 units in each dimension, and thus would convert this to an internal scale of 320 units per foot, thus assuring that the overall database was about 64,000 internal units on a side. Although the user need not specify a scale, the system is capable of computing one given the limits of the database. All dimensions can be computed and displayed in real-world units for the convenience of the user. Disadvantages of this approach include the fact that extending the boundaries of the original database becomes impossible. In addition, scaling the drawing to fit a given size of output is by no means likely to yield a drawing at standard architectural or engineering scale.

Whatever the position taken on this issue, the user of an IGDIM program can be saved a good deal of surprise and conceptual confusion if the nature of the "scale" of a drawing is explained in the documentation early and often.

3.1.3 What provisions are made in the program for digitizing existing two-dimensional data, including scale, rotation, and "edge alignment" procedures?

Discussion

All graphic input to IGDIM programs can be placed in one of two classes: sketch input or traced input. The user who is sketching into the database is not making tracings of existing sketches or drawings. Trace input is the operation
of placing an existing document on a digitizing tablet and using the cursor device to trace from the document into the database. If the user wishes to enter data already recorded on paper into the database, some provision for trace input must be made.

Interpreting traced input is by no means a simple matter. Many undesirable distortions can occur during this process. Two distortions for which accommodation must be made are rotational and edge matching distortion.

1) Rotational distortion: Both the source document and the digitizer have X and Y axes. If these are slightly misaligned, vertical and horizontal lines on the source document will not be traced into the database as vertical and horizontal lines. Among other problems, the resultant graphic aliasing of the slightly off-vertical and off-horizontal lines in the database will be extremely disturbing to the architectural user. Therefore, provisions must be made in the digitizing software such that the user can specify a horizontal line on the source document and the system software automatically rotates all entered line segments by the amount necessary to rotate the traced horizontal line into a horizontal line in the database.

2) Edge matching is another major problem in trace input. Large digitizing tablets are delicate, expensive, and require a great deal of space. The smaller tablets (i.e., 12" x 12") used in many design schools for graphic input are not large enough to permit tracing a typical architectural source document (24" x 36" or 36" x 48") in a single piece. Accordingly, large source documents must be cut into segments smaller than available digitizers and traced in pieces. The "edge matching" or alignment problem is that of putting the pieces back together again in the database.

For an IGDM program to permit adequate trace input, some provision in software must be made for rotational correction and edge matching of digitized input. It is not possible to use trace input if cursor keys are used as the graphic input device. Mice are not useful for trace input because 1) there is no "hot spot" - a single point on the device corresponding to a point on paper and 2) mice are relative positional devices rather than absolute. Unless they are held always in one orientation on a surface and never rotated, they cannot serve as digitizers. Digitizing tablets may or may not be used for trace input, depending on whether the programs using them have rotational and edge matching correction software.
Conclusion

The report summarized in this presentation represents an attempt to raise issues, not answer them. As part of the IBM AcfS project of which this report is a small part, we tested and reviewed more than 80 programs, 10 of which were commercial drafting and design database management programs. Testing these programs indicated that many different approaches to the design of the user interface can work, and work well. Dogmatic adherence to any single set of criteria concerning the design of the user interface is counterproductive. What is important is that as authors, teachers, and now consumers of programs for the design education community, we explicitly address these issues of the user interface before writing or purchasing systems for use in design education.
References

Note: The final titles for the documents referenced in this report were not yet set when the report was written. Therefore, these working document titles may change, but the report numbers should remain the same as shown here.


167


