# DESIGNING A GRAPHICS APPLICATION INTERFACE

John L. Wilson, Stuart S. Chen, Steven L. Sadofsky

Department of Civil Engineering, CAE Laboratory, Lehigh University
Bethlehem, Pennsylvania 18015

## ABSTRACT

This paper describes a systematic approach that can be followed by developers of educational courseware which contains graduated learning capabilities to accommodate a diverse group of users. The paper first presents and discusses a unified conceptual model of a computer-integrated engineering system where graphics plays an important function in the user interface. Second, this paper addresses an overall approach to be taken to interface design and the role graphics can assume. The development of an interface is then examined from both the viewpoint of the user and the courseware implementor. Computer generated displays are included which illustrate the use of the Graphical Kernal System (GKS) in picture generation and modification. The examples are chosen from interactive structural engineering courseware developed by the authors at Lehigh University.

KEYWORDS: computer-integrated systems, man-machine communication, interface design, courseware, structural engineering.

## INTRODUCTION

Many engineering applications can be enhanced by the use of a Computer-Integrated Engineering System (CIES) wherein the components of graphics, analysis/design, and data management are interrelated and implemented into a single, unified system. The concept of a "total design" via interrelated subsystems is well-suited for a wide variety of problem-solving stiuations in engineering whether they are oriented toward education, research, or practical applications. Among these areas there are common aspects of problem-solving that need to be addressed if we are going to use computers effectively to help us improve the way we look at problems as well as the way in which we solve them.

The content of this paper is focused on two major headings related to a computer-integrated engineering system. First, a conceptual, logical model of such a system is presented. This model is suitable as a framework for discussing the achievement of coordination and continuity of problem-solving tasks within a unified system. Second, the central issue of developing an interface within such a system is examined from both the viewpoint of the user and the program implementor. Technical specifics are discussed and relevant graphics displays are presented from structural engineering courseware developed at Lehigh University. Appendices are included to illustrate procedures for processing interactive replies, and for data input and modification.

## CONCEPTUAL MODEL

A conceptual model of a computer-integrated engineering system which illustrates the major components and their relationships is shown in Figure 1. The model is presented here to help clarify the complex set of interrelated processes that take place in the overall CIES. It will also be useful for later discussions on the details of the interface design considerations as well as the capabilities of particular subsystems.

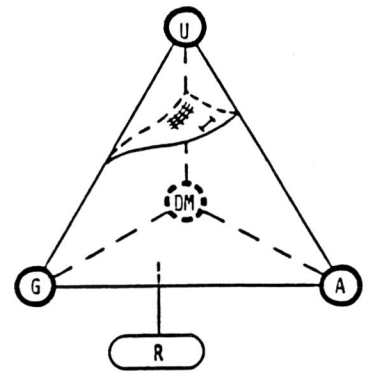



Figure 1 Conceptual Model of a Computer-Integrated Engineering System

As represented in this figure, the user (U) communicates with the subsystems of graphics (G), analysis/design (A), and data management (D) through a human-machine interface (I), shown by the shaded surface. The information produced, including interactive pictorial displays and/or tabular results is represented by (R). The

linkages for data transfer among these computer subsystems are indicated by the base plane of the model.

The generality of the above model has been presented and discussed elsewhere, Wilson, Fang (1984), Wilson, et al (1984). Its structure allows investigations of major tasks, how they relate to each other, and how information is routed during the program planning, design, and modification cycles.

The overall model contains a set of linked subsystems which function primarily as a whole. As mentioned, this model consists of three subsystems - graphics, analysis/design, and data management and their associated linkages. These linkages perform the task of transferring information, a process which is transparent to the user but quite relevant to the implementor. Though it is not the purpose of this paper to dwell on the details of data management or data linkages, it is worthwhile to review briefly the functional capabilities of the subsystems developed for the structural engineering courseware project presented herein.

The graphics subsystem, based on the Graphical Kernal System (GKS), contains the routines to display a variety of information and results to the user. The routines in the courseware, illustrated later in this paper, produce graphical representations of the structural components, external loadings, and support constraints. These displays allow the user to visualize and conceptualize the entire structural arrangement. All structural response quantities such as shear forces, bending moments, and deflections are displayed interactively in a more understandable fashion than that obtained from traditional engineering software. In addition, interactive displays of help facilities, and explanations of items such as program capabilities and sign conventions permit a greater degree of interaction and facilitate a better understanding of the courseware.

The analysis subsystem contains a set of procedure-oriented routines which perform an analysis based on the direct stiffness method for framed structures. Analysis data is retrieved from the data management subsystem, processed, and then returned for later use by either the graphics or the analysis subsystem. This incorporates flexibility for the user to perform and visualize "customized sensitivity studies" on what-if scenarios, e.g., selectively changing any parameters such as geometric configuration, material properties, loading conditions or boundary constraints.

The data management subsystem allows the user to create, maintain, and modify graphics and non-graphics data. Once a relationship is established between these two, the databases will appear to the user as a single database. Any data modified or updated is then reflected in the appropriate databases in the system.

Of particular interest in this paper is the interface mechanism since its versatility determines, to a great extent, the effectiveness and usefulness of a given system. The interface, properly designed, should support "customized" communications which allow a user to select from a variety of options among these subsystems.

## INTERFACE DESIGN APPROACH

This segment of the paper is centered around two basic sections. First, this section contains an overall approach and the role of graphics in the interface design. In the next section, the implementation of an interface is examined from two points of view - what the user desires, and then what approach was taken during the implementation for the problem domain at hand.

### Overall Approach

Interface design considerations are central to what is important in determining the usefulness of application courseware. The development of an interface which supports a two-way transfer of information along with the ability to explore what-if scenarios is the key issue addressed in the remainder of this paper.

As mentioned, the versatility of the interface determines the effectiveness of a given computer-integrated system. The characteristics of problem-solving, of course, preclude the development of a generalized or universal interface. Yet the idea of providing a flexible but not chaos-inducing interface, though difficult to achieve, is a most worthwhile pursuit. It is quite evident that the software generally available today is somewhat less than desirable for educational or learning purposes. Interface design techniques are still at an embryonic stage of development - similar to the state-of-the-art of device independent graphics of more than a decade ago.

In an attempt to highlight some of the fundamental considerations in the overall approach to interface design, the following three items are presented.

An effective interface should:

a) Let the user become more the controller of the application study and logic flow. Encourage experimenting and tinkering, since that is what engineers do.

b) Help develop a user's logical knowledge, and encourage his intuitive process - both can be aided with the use of graphics, Monk (1984).

c) Be more of a user information system, i.e., accommodate different expertise, and need to know levels. It should clarify, not confuse, in the user's learning process.

The authors recognize that these considerations may appear quite general and difficult to achieve. Though this brief paper cannot devote sufficient discussion to these broad items, more specificity is given to the implementation approach taken in a modest attempt toward their attainment in a specific problem domain and with "limited" user options for educational purposes.

In attempting to accomplish these lofty goals, a modular approach to program design and coding is mandatory. Briefly, this approach may be summarized as follows:

a) Identify the smallest "blocks" of the analysis subsystem (A) and data manipulation (D) that will occur during problem-solving scenarios. Then, modularize accordingly.

b) Identify specific typical interactive sequences that arise, and modularize the logic flow accordingly. A diagram depicting input and modification of data during one such typical dialog sequence appears in Appendix B - Procedure for Data Input and Modification.

c) Design a strategy to accommodate interactive data modifications and the "ripple effect" consequences to other data. That is, develop a strategy for keeping data up to date.

This modularity keeps the analysis subsystem (A) manageable, thus making it feasible to accommodate enhancements to (A). The corresponding enhancements to the interface (I), however, are much more difficult to accommodate, Wilson and Shull (1984).

Role of Graphics

While a carefully thought-out interactive dialog is very important, people do not think only in terms of words and phrases.

The courseware designer needs to understand, for the domain of interest, what concepts, help information, and results (output) can be expressed effectively in pictorial form. It is also worthwhile to examine how the use of graphics can possibly reflect the way practitioners think during problem-solving. The domain of interest in this paper, structural engineering, is well suited to this representation. The "experienced analyst ... always creates an image prior to analysis, but has become so familiar with the process that this critical step is no longer so readily apparent", Brohn (1983).

In fact, it may be argued that the thinking of structural engineers is essentially visual, Brohn. Physical phenomena of interest are conceptualized in visual terms, e.g., bending moment and shear diagrams, reactions, load vectors, supports (environmental constraints), and deflected shapes.

In this courseware, the specific uses of graphics are the following:

a) Review input data with the option to make changes. Add, change, and delete items such as joints, members and properties, supports, and loads.

b) Review output results in a convenient format, as an alternate to tables of numbers. Display forces and moments in the individual members as well as display reactions and deflections on the actual structure.

c) Provide help and explanation facilities to make answers to the user's questioning process more understandable in a visual sense.

The basic elements of the solution to a problem in structural analysis are straightforward physical relationships such as equilibrium, elasticity, and load-response theory. A typical difficulty faced by the student is putting these elements together properly to match the problem at hand. Once the visual representation is set forth, it becomes apparent that the numerical processing components (computations) are conceptually much simpler, Brohn. In fact in this particular courseware, the computations are done entirely in the analysis subsystem and are not seen by the user. The visual representation, then, attempts to encourage the development of intuitive knowledge of structural behavior, unencumbered by the computational procedures that often obscure a fundamental understanding of the structural behavior. Thus, the use of graphics here entails much more than mere graphing or plotting of data.

As stated, one goal of this courseware is to help users develop a "feel" for structural behavior. To accomplish this, the use of graphics is intended to affect the way the user thinks. This requires active thinking, questioning, and tinkering by the user so he is not just a passive viewer of "what-if" games.

## INTERFACE IMPLEMENTATION

As has been stated, what the user would like to see should be a major input to the process of designing the interface. What the user would like to see is not incidental.

What the user would not like to see or be bothered with is also important here. In the

problem domain herein, the inner workings of the (A)-(D)-(G) subsystems are transparent to the user. That is, the interface serves as a "shield" between the user and the computations. All courseware assumptions, limitations, and capabilities are, however, clearly defined and displayed for the user. He should be able to work through the program without unduly having to translate his thought processes into a rigid data format required for input to courseware.

While significant attention should be paid to what ideally characterizes the user interface (from the user's point of view) one must grapple with how each aspect is actually to be accomplished in the program design and coding. This provides the rationale for the arrangement of topics in this section.

Under each topic, the desires of the user are mentioned first. Then, the approach taken to implementation is described. In each case it will become evident how the idealism of user desires is tempered by practicality, while not being forgotten.

Natural Language

Much has been written about the desirability of using a natural language in problem solving. It is no panacea, however. There are many useful references on the development and uses of natural languages, Sowa (1984).

Some of the basic features a user might desire are summarized as follows:

What the User Desires:

Use of a common, flexible framework for expression in a natural language (not necessarily a completely unrestricted language).

Use of a language which emulates conversations between people (not necessarily one which accepts a full range of a human language).

Provision of feedback for help, detecting errors, and clarifying ambiguities (not necessarily unlimited assistance).

Implementation Approach:

The implementation approach consisted of developing the following capabilities:

Use of a language level in between a computer language and a natural level language. The intermediate notation is expressive enough to capture the English meaning and also have a formally defined mapping to the logic flow of the courseware.

Use of keywords, not a full natural language. Often a linear string of keywords is preferable to making multiple menu selections.

Interactive use of commands (action), queries (where am I), and help facilities (what is the meaning of) to encourage a flexible dialog. The courseware remembers what has been chosen and what is the present topic. It also interprets user responses.

Limited feedback for detecting errors - based on anticipating specific input errors and building in appropriate error messages.

As yet no program can accept the full range of language and interpret all the nuances and shades of meaning that a person might ordinarily use, Sowa (1984). It is the intent of this courseware to permit the user, in a limited extent, to develop a dialog pattern using the vocabulary normally encountered within the problem domain at hand - structural engineering.

Dialog Pattern

What the user wants to see depends very much on how familiar he already happens to be with the use of the program. For the sake of simplicity here, users are categorized as beginning users and as experienced users, ignoring the broad spectrum in between. What these two categories of users would like to see may be summarized as follows:

What the User Desires:

The beginning user typically wants and needs frequent prompting, in order to accomplish something useful without difficulty, and without having to know too much, Chen and Wilson (1985). He does not necessarily want to know all choices available at each particular stage, just an adequate choice. For such a user, complex menus can be intimidating. He is probably not yet interested in doing much tinkering, preferring to develop a certain level of confidence in his use of the program before experimenting with other options.

The experienced user, on the other hand, may become bored or frustrated with frequent prompting and with a pre-set course of action. He generally knows what he wants to do and wants to do it in a minimum of time. He would like to have the flexibility to bypass some prompts entirely and tinker with the problem at hand. In fact, the tinkering instinct is not at all inconsistent with the educational goals of the courseware.

Implementation Approach:

From the above discussion, it is evident that what is "user-friendly" to one user may be quite "user-oppressive" to another. The implementation approach needs to take into account at least the extremes in user familiarity levels.

The steps taken may be described as follows:

1. Decide on the format of the interactive dialog. Possibilities considered in the design of the courseware were: menu selection; commands entered by the user; and limited option prompting with a forced sequence for beginners.

   The format chosen is the limited option, "forced" framework, which may be viewed as a default "tinkering" pattern. The main reason for this decision is the relative feasibility of accounting for well known human factors considerations, see Foley and VanDam (1982). In these regards, consider the following:

   - Limited user options are easier to accommodate, since fewer possibilities and branches need to be built into the courseware design and coding.
   - Simple, consistent interaction sequences are easier to design, while providing appropriate feedback to the user.
   - Graceful error recovery is more attainable, since there are few correct inputs at any given stage of program execution.

2. Provide acceptance of any of a limited amount of out-of-sequence global commands entered by experienced users within this framework. In this way some provision is made to accommodate different user knowledge levels. Also, some explanatory messages are omitted from the "forced" sequence when the user has declared himself as an experienced user.

Throughout, the goal is to enable the user to construct and modify the objects (entities) being dealt with in the program, interpret the results, and proceed accordingly.

Graphics

The courseware goal of interest here is to develop a "visualizing" capability as discussed earlier.

What the User Desires:

The user would like to see the effect of his specifications and modifications on the structure topology, structure loading, and on the structure response. He generally wants the capability of selective display of any of these items.

Implementation Approach:

The implementation approach taken for this courseware includes the following six steps:

1. Define an application model of objects and their relationships. Also identify and account for properties required of the analysis subsystem (A) in Figure 1.
2. Identify all the data used in (A). This information is needed in step 6 listed below.
3. In light of graphics hardware and software capabilities, lay out each display. Design the screen layout to accommodate interactive dialog and the display of results. The accounting for user desires consistent with courseware goals is especially important here.
4. For each display, decide on graphics primitives and attributes to employ in the display. This will not always be self-evident. For example, the use of different text fonts may be desirable when a program may occasionally be used on a low-resolution terminal. As another example, the segments that may be modified interactively need to be made "PICKable".
5. Define windows and viewports for maximum flexibility. Windowing also helps in picture definition and in structuring the displays.
6. Identify links between graphics data (data needed to create the displays) and existing analysis subsystem (A) data. For example, consider the following situations:

   - Use some analysis (A) data as it is. For example, joint coordinates in the analysis model are used directly in plotting the structural topology.
   - Adapt some analysis (A) data. For example, joint coordinate data is scanned for maximum and minimum values to compute the window coordinates for the display showing the structure itself.

- Create some entirely new data. For example, unique segment names need to be assigned to each segment.

It should be noted that the above steps are not necessarily chronologically arranged. Indeed, some iteration and modification of each step will probably prove necessary during courseware design and coding.

Help Facilities

Users in general would like to have relatively effortless access to information characterized by relevance, clarity, and brevity all at once. In the ideal case, this information would be geared to his current familiarity level and deal with the specific situation giving rise to the current request for help, Chen and Wilson (1985).

What the User Desires:

For beginning users, help requests seek information on the following:

a) Concepts and terminology used in the domain or courseware

b) Explanations of what is being prompted for and why; and

c) Assistance to get through the courseware, unscathed, and arrive at results.

Experienced user help requests typically seek information on topics such as:

a) Current options; and

b) Analytical assumptions of the courseware.

The help needs depend to a great extent on the knowledge level of the user.

Implementation Approach:

At the very outset, it is recognized that it is impossible to provide what is desired (described above) in a general way. There is no way to know a priori what help a particular user needs.

In spite of these very serious caveats, an approach is taken which retains some desireable features of a reasonable help facility. Keeping in mind that we do not want to bury the user with information, first identify the categories of help requests likely to be entered by either beginning or experienced users. Each category is to contain a digestible amount of help information.

What is provided here is two different help facilities, one for beginning users, one for experienced users. Each tier of help handles a different category of help that a beginning or experienced user, respectively, is likely to need via a 'query-in-depth' scheme, Gaines (1981).

When a user enters '?' or 'help' requesting help, a message appropriate to the first tier or category appears. If that message does not satisfy the user, he can request further help by again entering '?' or 'help'. Courseware control then passes to the second tier, with a help message on a different topic.

The particular tiers employed in this courseware are as follows:

A) Beginning User:

Tier 1: What is being asked for and why

Tier 2: What you just did, and what you are currently doing

Tier 3: Current input options and consequences

Tier 4: Program assumptions, capabilities, limitations

B) Experienced User:

Tier 1: Current input options, including global commands

Tier 2: Program assumptions, capabilities, limitations.

It was decided to accommodate help requests or global commands at any point in the interactive dialog where a yes/no reply from the user is sought. In this way, it was possible to design a master subroutine to process all replies entered by the user during the interactive dialog. It is noted that help requests and global command entries are not permitted during interactive data entry. A flowchart describing the procedure to process interactive replies is given in Appendix A.

## REPRESENTATIVE DISPLAYS

As previously described, one of the uses of graphics in this courseware is to facilitate the change and modification of structure topology and boundary conditions and investigate the effects of such changes on load carrying capacity and structural response. Due to space limitations, only one example is presented here. Figure 2 shows a bridge span as it might be modeled during initial design. The small triangle represents a fixed support bearing and the small circle represents a roller (expansion) bearing.

Consider the situation where the user wishes to investigate the structural response to given

loads for the case where a bottom chord member has failed. Figure 3 shows how the structure has been modified to model this case. The failed member has been removed, and the roller has been replaced by a fixed support, representing additional restraint contributed by, say, the bridge abutment.

Performing a structural analysis for the two configurations and comparing the results would reveal fundamental differences in the behavior of the structure, in this case due to an "arching" effect in the second configuration. The use of graphics readily enhances the user's ability to develop a "feel" for these kinds of variations in structural response.

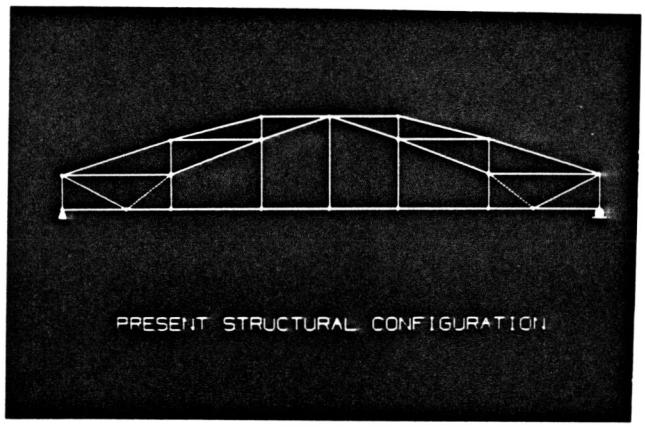



Figure 2 Initial Arrangement

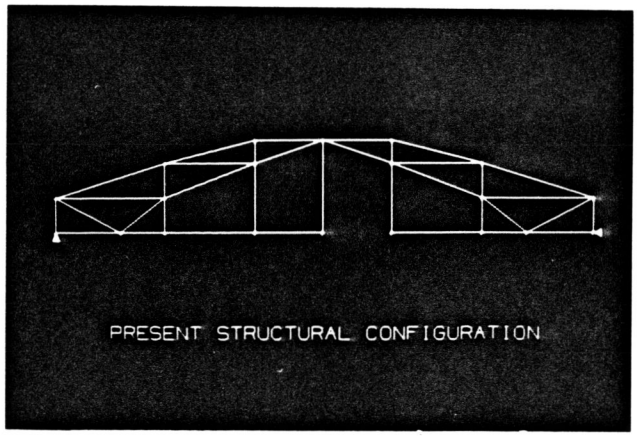



Figure 3 Modified Arrangement

SUMMARY

The major thrust of the civil engineering courseware being developed in the CAE Laboratory at Lehigh is to improve the way users look at problems as well as the manner in which they solve them. Toward this end, a conceptual model was presented which includes an interface between the user and a computer-integrated system consisting of graphics, analysis, and data management subsystems. Then, the central issue of developing an interface to accommodate different levels of user knowledge, in a given problem domain, was examined, with specifics, from both the viewpoint of the user and the program implementor. In conclusion, a rational and manageable approach to interface design for engineering courseware has been presented.

REFERENCES

1. D. M. Brohn, "Academic Priorities in Structural Engineering - The Importance of a Visual Scheme", Structural Engineer, Vol. 61A, No. 1, Jan. 1983.

2. S. S. Chen and J. L. Wilson, "The Generation of User Oriented Documentation", Proc. Second Int'l. Conf. on Computing Civil Engineering, Hangzhou, People's Republic of China, June, 1985.

3. J. D. Foley and A. Van Dam, Fundamentals of Interactive Computer Graphics, Addison-Wesley, 1982.

4. B. R. Gaines, "The Technology of Interaction - Dialogue Programming Rules", Int. J. Man - Machine Studies, Vol. 14, No. 2, Jan. 1981.

5. A. Monk, Fundamentals of Human Computer Interaction, Academic Press, 1984.

6. J. F. Sowa, Conceptual Structures, Addison-Wesley, 1984.

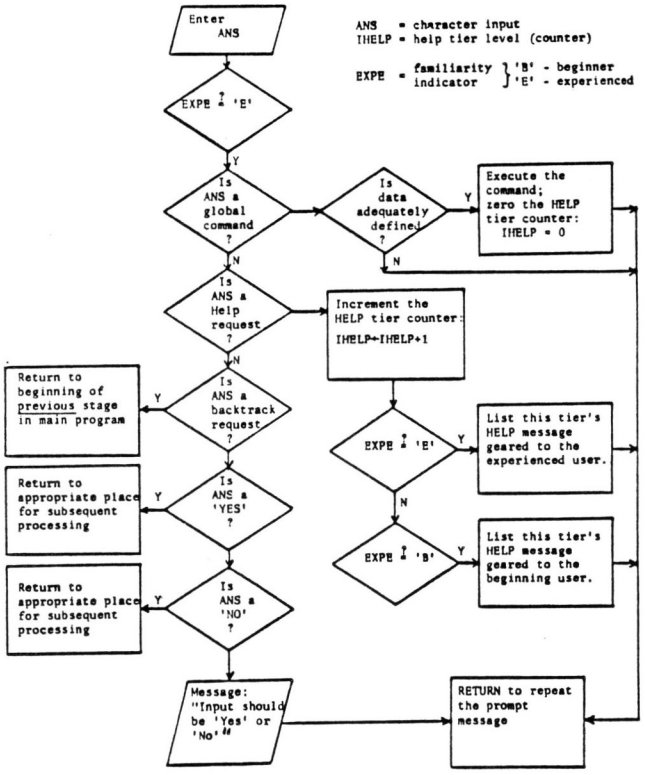
7. J. L. Wilson and H. Y. Fang, "Computer Graphics Applied to Construction Processes in Offshore Structures", Int'l. Conf. Proc. for Computer Aided Analysis and Design of Concrete Structures, Split, Yugoslavia, Sept. 1984.

8. J. L. Wilson, G. K. Mikroudis and H. Y. Fang, "Expert Systems Application in Environmental Geotechnology", Proc. Conf. on Computer-Aided Instruction and Research, ASEE, Arizona State Univ., Oct. 1984. (Academic Press)

9. J. L. Wilson, B. A. Shull and J. N. Gregus, "Process Plant Design Using Interactive Graphics, ASCE Proc. of the Third Conference on Computing in Civil Engineering, San Diego, CA, April 1984.

APPENDIX A

PROCEDURE TO PROCESS INTERACTIVE REPLIES




APPENDIX B

PROCEDURE FOR DATA INPUT AND MODIFICATION