The Delicate Balance: Introducing Complex Software While Teaching the Discipline's Concepts

Timothy J. Sexton, Ph.D.
Ohio University, Athens, Ohio

Introduction

Several disciplines have been revolutionized by the development of very powerful yet complex computer software. Statistics and Engineering Graphics are just two examples of subjects in which complex software can assist the understanding of difficult concepts and allow students to be more productive. But along with this revolution in software, there comes a perplexing problem. How does one balance the teaching of the basic concepts of a discipline and also teach complex software? If you concentrate on the software, students may become proficient but dangerous software users. They will not know its proper applications or the reasonableness of their solutions. If you concentrate on basic concepts at the expense of the software, students are not being exposed to the most efficient and effective tools of the respective discipline.

The author teaches courses in Engineering Graphics in which Computer Aided Design (CAD) software is an essential and integral component of the discipline at the entry through advanced levels. But CAD software can be overwhelming to the student using it for the first time. Add to this overwhelming feeling the expectation that one must learn the software while simultaneously trying to master the basic concepts of Engineering Graphics. This paper discusses the author's experience using CAD software in an introductory course in Engineering Graphics to illustrate the problem of introducing a complex computer software in an entry level course. Instructional methods for successfully implementing complex software into an entry level course will be suggested.

Problem Statement

Engineering Graphics courses have been taught for some time without complex software, so why work so hard to implement it into the curriculum? Research by Bertoline, and Sexton compared the use of both 2-D and 3-D CAD in contrast to traditional manual and 2-D CAD approaches in teaching Engineering Graphics and Descriptive Geometry, They found no significant difference on test scores measuring spatial visualization of students using traditional methods versus CAD. This implies that the use of CAD in Engineering Graphics courses is at least as effective in teaching spatial visualization.

This author works under the premise that CAD software is more than just a means to produce drawings more efficiently. Its three dimensional capability allows students access to an environment never before possible, This complex software allows students to better learn and reinforce the basics of Engineering Graphics ! It is worth the effort to implement it into the curriculum. However, due to its complexity, it must be implemented properly or it can be disastrous to your students.

- So how complex is "complex software"? To illustrate how complex a software can be, Figure 1 shows the number of manuals and pages that come with a typical CAD software. Keep in mind that each student will also have a theory text of about 600 pages and many drawing assignments. You can see that the addition of CAD software to the first course in Engineering Graphics makes the course appear overwhelming.

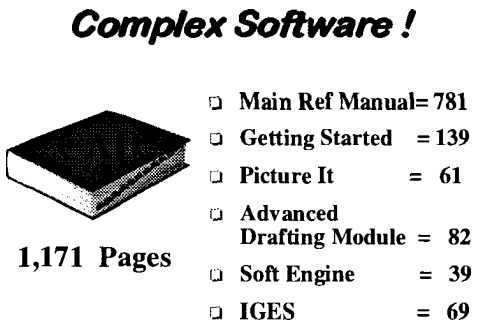

Figure 1: Number of manuals and pages in a typical CAD software

Proposed Solution

This author proposes that the software be taught first without introducing the basic concepts of Engineering Graphics. This will help relieve student anxiety and frustration by allowing students to become comfortable with the basic commands and "logic behind the software", without needing to worry about studying and applying concepts of Engineering Graphics. Once comfortable with the basic feel of the software, the basic concepts of Engineering Graphics can be introduced and the three dimensional aspects of the software can be used.

Critics of teaching software by itself let alone teaching it before the concepts, believe: 1) teaching software is comparable to remedial work and students should bear the responsibility of picking it up on the own time, **2)** software is simply a tool, and should not be emphasizes or singled out in any way; and/or 3) the teaching of software cannot be separated from the concepts of the related discipline.

Teaching software first has many advantages to both student and instructor. It allows: 1) the time beginning students need to overcome the fear of what appears to be an overwhelming software; 2) students to become comfortable using the software before needing to learn concepts of Engineering Graphics; 3) the instructor can control what portion and in what order the students will be exposed to the software.

The author suggests that the software be introduced using step-by-step tutorials. The tutorials should provide detailed instructions and adequate illustrations so the student can see what their drawing should look like at different stages. The tutorials should begin with the exploration of the working environment and file management, and then explore the basic commands. The tutorials should be designed to ensure students are exposed to all the basic commands. For example, students should be exposed to all the methods for creating lines and circles. This approach is significantly different from most tutorials that come with CAD software. In manufacturer supplied tutorials, students are guided through a project whose end results "look" more impressive, but the lack of breath in the basic commands is detrimental when more complex drawings are encountered.

Tutorial Design

Inherent and critical to the success of this instructional method is the need for the tutorials to follow sound instructional design principles. Some of the important principles imperative for success are: 1) stating

what the task is (including the commands introduced); 2) providing a reason for completing the tutorial, logical sequencing and flow within and between tutorials; 3) adequate number and quality of illustrations; 4) a simple command statement format; 5) anticipating and heading off problems by providing helpful hints (based on the instructor's experience and iterative review process). Figure 2 illustrates a possible format by listing the major heading titles and purpose for each section used for the tutorials.

Headings	*Purpose*
Title:	Topic
Focus:	What to do & commands explored
Rationale:	Why do it
Objectives:	What you'll be able to do after the tutorial
Instructions:	Step-by-step
Points to Remember:	Special hints

Figure 2: Tutorial's Format

To illustrate how detailed the tutorial needs to be, Figure 3 provides a small section of a tutorial on drawing tangent lines. Figure 4 is one illustration that would accompany this section of the tangent lines tutorial. A series of progressively more complete illustrations are used to help a student judge their progress.

Creating Tangent Lines

Problem 1A

1. Select **1 CREATE, 1 LINE, 4 TAN/PRP** (tangent/perpendicular), 3 TAN TAN (tangent tangent).
2. Prompt *"Indicate 1st tangent entity. "*
3. Click on an outside left edge of circle 1.
4. Prompt *"Indicate 2ne tangent entity. "*
5. Click on an outside left edge of circle 2.
   - Check your work with Figure 9-17A.
   - If the line created is not correct, use **Ctrl-Q** to delete the line and try again.
6. Repeat the previous steps to create a tangent line to the right outside edges of circles 1 and 2 as illustrated in Figure 9- 17B.
7. Repeat the previous steps to create two tangent lines to that cross between the circles 1 and 2 as illustrated in Figure 9- 17C.

Figure 3: Section of a tangent lines tutorial

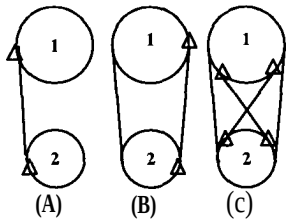

Figure 4: Illustration (Figure 9- 17A, B, C) used in a tutorial on drawing tangent lines

Using step-by-step guided tutorials has proven to be successful but there are some drawbacks: 1) tutorial are very time consuming to develop and need to be updated every time there is a change in the software; and 2) since all students are doing the same thing, there is ample opportunity to copy someone else's work. A common, but debatable, criticism of using step-by-step tutorials is that you are simply spoon feeding students and they are just following directions and not learning the logic behind the software.

In the author's experience the advantages far outweigh the disadvantages. The advantages of using step-by step tutorials are: 1) allows students to work at their own pace; 2) allows directed work to go on outside supervised labs; 3) students are guided through the software in an organized and logical manner; 4) you can be guaranteed that each student is exposed to a number and variety of software tools; 5) students get to work with and have a reference of how individual commands work together and are interrelated; and best of all 6) the instructor has time to answer quality questions instead of spending time answering "which button do I push" type questions.

Conclusion

Because complex software is so time consuming to teach and use in a course, it should only be used if it enhances the learning of the basic concepts of the material. If it is used, you must be sure that students don't get frustrated because they are trying to learn the software while the instructor is charging ahead introducing new theoretical concepts. The author has found the statement "you can pay me now or pay me later" to be very appropriate when using complex software in a course. When CAD software was integrated with theory, students would learn just enough software commands to get a specific problem finished. Then when more difficult problems are assigned they try to draw them with a limited bag of software tools. This lead to much frustration because they must go back and learn additional software commands to make the more difficult assignments manageable.

To help avoid student frustration, the author proposes that the basics of the software be taught first before introducing theoretical concepts using step-by-step tutorials. Once students feel comfortable and begin to see the "logic behind the software", theory can begin to be introduced and additional software commands can be introduced as they are needed to support the theory. Most important to this author is the opportunity to get students excited about the subject matter. Software should enhance this opportunity not prevent it !

References

Bertoline, G. R. (1987). The Effect of Using CADD (computer-aided design drafting) to Learn Engineering Design Graphics. *Dissertation Abstracts international,* (University Microfilm No. 8717604).

Sexton, T. J. (1991). Teaching Engineering Graphics: A Comparison Between Manual/ Two Dimensional Computer Aided Drafting Traditional Methods and Three Dimensional Computer Aided Drafting Nontraditional Methods with Respect to Spatial Visualization Ability. *Dissertation Abstracts International,* (University Microfilm No. 9220479).