Teaching GUI in Operating Systems Courses

Lior Shamir
Michigan Tech
1400 Townsend Dr.
Houghton, MI
lshamir@mtu.edu

ABSTRACT

Graphical User Interface is an integral part of many modern operating systems, and GUI-based features have been attracting considerable attention from operating system developers. However, while OS concepts such as CPU scheduling, memory management, disk management, file systems and security are typically covered in operating systems courses, GUI is poorly studied in colleges in the sense of operating systems. In this paper, a section discussing GUI topics in an operating systems course is described. Due to the increasing popularity of GUI components as integral parts of modern operating systems, we suggest to include a GUI section in undergraduate level operating systems courses.

Categories and Subject Descriptors

K.3 [**Computers and Education**]: Computer and Information Science Education—*Curriculum*

General Terms

Keywords

Operating Systems, GUI, Graphical User Interface

1. INTRODUCTION

Since the early 1990's, operating systems that include an integral graphical user interface have been becoming increasingly popular. This can be reflected by the success of Microsoft's Windows and Apples's MacOS. The ease-of-use introduced by GUI operating systems is an important advantage over traditional command-line systems that allows non-computer experts to access and effectively use computing machines. In fact, the ease-of-use and enhanced accessibility of GUI operating systems is considered by many as one of the most influencing technologies in the history of computer systems, and is often associated with the personal computer revolution [3, 4]. Traditional command-line operating systems such as Linux have also been enhanced by GUI features.

However, while the importance of effective graphical user interface as an integral part of modern operating systems has been becoming increasingly important, this topic has yet seem to attract just little attention in undergraduate operating systems courses, and many of the popular textbooks such as "Operating System Concepts" [9] and "Modern Operating Systems" [10] do not cover this topic.

Some schools indeed offer courses of graphical user interface design. However, these courses are usually focused on topics of human-computer interface, rather than discussing the internal mechanisms that enable these interfaces. GUI design courses are not as common as introductory operating systems courses, and are usually offered as electives. Therefore, in many colleges, computer science students can graduate without taking a GUI course.

In this paper we propose to consider the inclusion of graphical user interface in the curriculum of undergraduate level operating systems introductory courses (e.g. "Introduction to Operating Systems"). The paper describes the GUI section that covered in the course "Introduction to Operating Systems" in the summer semester of 2006. In Section 2 the computing environment used for the course is described, in Section 3 the course details are discussed, and in Section 4 the student evaluation of the section is presented.

2. THE COMPUTING ENVIRONMENT

The computing environment selected for the GUI section of the course was Windows XP with Visual Studio 2005. The main reasons for selecting Windows XP were its powerful integral GUI and its popularity. Since Visual Studio is the most popular development suite under MS-Windows, it seemed natural to use it in the course. Although using just a small part of this software suite in the course, working with the newest version of Visual Studio was considered by many students important.

Like some other schools, Michigan Tech typically gives a relatively low priority to programming in MS-Windows environment, so that students earn just a little experience in Windows while studying toward their CS degree. However, this approach does not always meet the industry demand for knowledge and experience in some common programming environments [5]. Therefore, this section of the operating systems introductory course also provided the opportunity to introduce students to programming in this popular environment. As explained in Section 4, students reported that they believed that the Windows programming section was the most practical section of the course, and also rated it as the most likely to be used in real-life programming.

Since Windows labs are typically smaller than UNIX labs, the number of students registered to the course should be taken into considerations when opening the course. The case described in this paper was a summer course with only 11 students, which provided a friendly environment for experimenting this new topic.

The programming environment was Microsoft Visual C++, which is included in the Visual Studio 2005 suite. Although some students indicated they were familiar with Visual Studio, no student in the class actually had programming experience in that environment. Mastering Visual Studio is not beyond the capabilities of the average CS college student [6, 7], but a learning curve still exists. In order to allow students to compile and run their projects without spending too much time studying Visual Studio, students were provided with a detailed explanation of how to start, compile and debug a simple consol application in Visual C++. Although some students reported that a significant portion of their time was consumed by attempts of starting and compiling their projects, all students managed to use Visual C++ effectively.

3. GUI SECTION DETAILS

The purpose of the GUI section in the operating systems introductory course is to discuss basic concepts of graphical user interfaces in the sense of operating systems, and 3 one-hour classes were sacrificed for studying the topic. The section included system components, resource allocation, window painting and handling, and application programmer interface. The course did not include topics related to GUI design such as usability and ease-of-use, which are typically discussed in User Interface Design courses.

The section started with a general description of graphical user interfaces and their role in modern operating systems, along with a brief history of GUI, starting from the mythological LISA to X-Windows and Windows XP. The purpose of this discussion was to provide an incentive for studying the topic by discussing terms students were familiar with from outside their CS college education.

The technical part of the course started with the description of the GDI (Graphical Device Interface) and *User* subsystems in Microsoft Windows. This included the purposes, scope and interface of each sub-system. The advantages of using GDI are explained in the light of the limitations of this design. The main downside of GDI is obviously the sacrifice of speed and its inability to support high-performance graphics, which results in poor animation and rendering capabilities, making it unsuitable for multimedia and computer games. Since many students are particularly interested in computer game technology [1, 2], the discussion also included solutions to the GDI performance problem in the form of DirectX or OpenGL, that provide faster access to the graphics hardware.

The next discussion focused on describing how windows are handled by the system. The discussion included window hierarchy, window positioning (X, Y and Z axes) the organization and use of the Z-order, window painting, parent and child windows, and pre-defined control windows provided by MS-Windows (such as *Button*, *Edit*, etc'). The discussion also included message transferring and a general description of the event-driven approach used by MS-Windows graphical user interface. The discussion was demonstrated by analyzing some simple situations. For instance, students needed to analyze which windows will receive a *WM_PAINT* message from the system, and by which order, given a specific event and window positioning. Students were also required to analyze situations of I/O events such as mouse movements, mouse clicks or keyboard events. The analysis included specifying which messages were sent by the system, and which windows received these messages.

The last discussion in the GUI section of the course was focused on the GUI programmer interface in MS-Windows. This discussion was a more pragmatic approach to the concepts studied in the previous discussion, and included window procedures, message processing, and basic Windows APIs such as *CreateWindow*, *FindWindow*, *SetWindowPos*, etc'. The discussion was demonstrated by short C codes of MS-Windows programs. These programs described basic Windows GUI features such as creating simple windows, creating control windows, and handling Windows messages using window procedures. Slightly more advanced features were retrieving and setting windows positioning (size, location and Z-order), and receiving messages triggered by I/O events. Programming assignments included programming a simple key logger that records a macro of keystrokes and mouse movements, and then plays the macro by simulating the mouse movements and keystrokes using the APIs *SetCursorPos* and *keybd_event*. Another simple programming exercise that attracted some student attention was a program that closes all open browsers using *FindWindow* and *SendMessage* APIs.

Programming assignments of other sections of the operating systems course that were given before the GUI section was studied required students to work both in Windows and UNIX. The idea of using more than one operating system was to let students learn how the same OS concepts are implemented in different operating systems, and to make the course as general as possible by detaching it from a specific operating system. Previous reports indicated that using Linux and Windows programming in the same course leads to a more inclusive and effective studying [8]. Therefore, by the time students started working on the assignment of the GUI section, they had some basic knowledge in programming using Visual Studio under Windows XP. However, since most students did not have any previous experience in Windows programming before taking the course, students were allowed to work on the programming assignments in pairs.

4. RESULTS AND STUDENT EVALUATION

An anonymous survey was given late in the semester (summer 2006) to students enrolled in the operating systems introductory course (CS-4411). In the survey, students were asked to evaluate each of the topics that were discussed in the course based on three questions. The topics were process scheduling, process synchronization, Windows multithreading, Windows GUI, memory management, inter-process communication, deadlocks, secondary storage devices, and interrupts. In their answers to all three questions, students expressed their particular interest in the GUI section. Few students commented on the large amount of work they had to sacrifice while solving the programming assignments. The questions and the survey results are the following:

1. In a scale of 1 to 5, how do you think each of the following sections contributed to your general understanding

of computer systems?

2. Do you think you will find the following sections of the course practical in real-life programming?

3. On which of the following subjects would you want to learn more?

Student answers to these questions are described in Tables 1,2 and 3.

Section	Average Student Evaluation (1 to 5)
Process scheduling	4.0
Process synchronization	4.3
Windows multi-threading	4.4
Windows GUI	4.2
Memory management	4.5
Inter-process communication	4.1
Deadlocks	3.5
Secondary storage devices	3.9
Interrupts	3.7

Table 1: Average student evaluation of question 1

Section	Average Student Evaluation (1 to 5)
Process scheduling	3.7
Process synchronization	4.1
Windows multi-threading	4.8
Windows GUI	4.8
Memory management	4.0
Inter-process communication	4.4
Deadlocks	2.5
Secondary storage devices	3.1
Interrupts	3.0

Table 2: Average student evaluation of question 2

Section	Average Student Evaluation (1 to 5)
Process scheduling	3.5
Process synchronization	3.8
Windows multi-threading	4.3
Windows GUI	4.7
Memory management	4.1
Inter-process communication	3.8
Deadlocks	2.9
Secondary storage devices	2.6
Interrupts	2.6

Table 3: Average student evaluation of question 3

The operating systems course described in this paper was a summer course, and therefore had only 11 registered students. However, it was noticeable that the discussions attracted more student attention than other sections studied in the course. This observation is also reflected by the results of the survey, and was the driving impetus for the creation of this paper.

5. CONCLUSION

Due to the increasing importance of graphical user interfaces in modern operating systems, we suggest to include a brief discussion of this topic in undergraduate level operating systems courses.

In the sense of operating systems, we suggest that the GUI section will cover the top-level GUI system design and components, window painting and handling, GUI messages, and programmer interface. Simple programming assignments were based on Visual Studio 2005 under MS-Windows XP.

The results of the student evaluation and the active discussions in the class indicated that students were particularly interested in the subject, and believed the subject was important both for their knowledge as computer scientists and for practical real-life programming.

6. REFERENCES

[1] K. Becker. Teaching with games: the minesweeper and asteroids experience. *Journal of Computing in Small Colleges*, 17:23–33, 2001.

[2] K. Becker and J. R. Parker. All i ever needed to know about programming, i learned from re-writing classic arcade games to be presented at future play. *The Intl. Conf. on the Future of Game Design and Technology*, October 2005.

[3] M. Dempsey. Jobs in the computer industry. *Nature*, 357:40–41, 1992.

[4] C. Depcik and D. Assanis. Graphical user interfaces in an engineering educational environment. *Computer Applications in Engineering Education*, 13:48–59, 2004.

[5] J. D. Fernandez and P. Tedford. Evaluating computing education programs against real world needs. *Journal of Computing Sciences in Colleges*, 21:259–265, 2006.

[6] T. Goulding and R. DiTrolio. Incorporating realistic constraints into a student team software project. *ACM SIGCSE Bulletin*, 37:54–58, 2005.

[7] B. Higgs and M. Sabin. Towards using online portfolios in computing courses. *Proc. of the 6th conference on information technology education*, pages 323–328, 2005.

[8] J. Phillips, J. and N. Ander. Design of a two-course sequence in web programming and e-commerce. *Journal of Computing Science in Colleges*, 19:208–217, 2003.

[9] A. Silberchatz, B. G. Galvin and G. Gagne. *Operating System Concepts*. John Wisely and Sons, New York, 2004.

[10] A. S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, 2001.