DEVELOPING SIMULATIONS AND EXPERIENTIAL EXERCISES ON THE PERSONAL COMPUTER: SOME CRITICAL ISSUES

Jerald R. Smith, University of Louisville

ABSTRACT

Nov that faculty and students are performing more of their computing activities on personal computers, the responsibility for understanding the technology has been placed in the hands of the user. This paper addresses some of the issues that the PC user confronts as they move from the computer-center's mainframe to the desktop PC.

Introduction

it is sometimes a bit of a shock to be reminded that, in operational and practical fact, the medium is the message. This is merely to say that the personal and social consequences of any medium result from the new scale that is introduced to our affairs by each extension of ourselves or by new technology..." (McLuhan, p. 390).

This famous quote is at the heart of a critical issue: developers and users of simulation games and experiential exercises are confronting a new medium in the personal computer. The pencil-and-paper exercise produced and distributed by a publisher and the mainframe business game that is controlled by a professional computer center have been translated to the personal computer.

One can infer that the message is intended to be similar even though the medium has changed. The medium change has profound effects on the exercises and games for both developers and users. This paper is directed at the implications and present-day consequences of the medium change that has led many developers, users, and students to question whether the exercise is to learn business principals, or to master the environmental turbulence of the personal computing medium. The critical issues surrounding the medium change involve hardware, software, data, and people. (Burch et al, 1983).

A Technical Statement of the Problem

The honeymoon with the personal computer and experiential techniques appears to be running into a mid-life crisis. In the past five years, much new simulation and experiential exercise development has been carried out on the personal (single-user) computer. The popularity of personal computer-based experiential exercises stems from several sources:

1. Classroom activities are spared the time-lags created by mainframe downtime or program re-runs, busy signals on remote-access lines, the wait for output produced by the computer center and other domain issues.

2. The user-interfaces are more "friendly" than the remote batch jobs created and run on terminals.

3. Personal computer exercises and simulations are a clear message to AACSB that computers are being integrated into curricula.

Although the early PC activities were limited by the need to use interpretive BASIC and memory (RAN) sizes, these constraints have been lifted as new interactive programming languages are developed for personal computers and as memory, size approaches mainframe levels. The brand and operating system incompatibilities that plagued the first three years of this decade are less pronounced as the industry standardizes on MS-DOS. However, these recent positive developments have created a set of problems of their own.

Problems with Hardware

Although the personal computer industry is beginning to standardize its hardware, academic software developers are still confronting hardware incompatibilities. These differences occur primarily in the output devices. For example, business graphics may be a desirable feature in a simulation game; however, developers are currently limited by the default configuration of bit-mapped video displays that produce less than adequate graphs. Most hardcopy output needs to be limited to a dot-matrix or daisy wheel printer that uses the Epson or IBM standard character set. Any assumptions beyond this can create output problems for the user.

Recently developed local area network technology (LAN) has encouraged many simulation developers to envision the "truly interactive game" in which the student or trainee players compete on-line. However, this technology lacks any standardization; at this writing the PC-based on-line game is a vision that awaits maturing of the LAN technology so that player files can be opened and merged successfully.

Some of the hardware technology problems have been mitigated with the advent of PC versions of third and fourth generation easy to learn programming languages that can be compiled to overcome the basic hardware incompatibilities. Languages such as QuickBasic[1] and Turbopascal[2] provide the ease of programming and the ability to generate hardware independent compiled code so long as the underlying assumption is that the user will possess an MS-DOS computer with a bitmapped display and a printer supporting the IBM character set.

As the basic configuration hardware problem becomes resolved, simulation developers like the authors are now confronting a new generation of problems to be considered. The 8088 microprocessor that emerged as the first-generation hardware standard is being rapidly replaced with faster processors and different disk technology. The quadruple-density disks drives that come with the new generation of PC's (e.g. IBM-AT) can read the simulations and exercises developed for the original drives; however they cannot write "readable" data back onto the disks. This means that the disks that are distributed with the exercise must receive special preparation unfamiliar to the developers and users in order to be usable.

[1] QuickBasic is a product of Microsoft Co.

[2] Turbopascal is a product of Borland Co.

Exercises such as Schreier and Dondlinger's Electronic In-Baskets (1986), and simulation games such as Micromatic (Scott and Strickland, 1985), Airline (Smith and Golden, 1986) and Stratplan (Hinton and Smith, 1985)[3] are confronting this emerging hardware problem that needs to be addressed.

DOS Program

The DOS (disk operating system) developed by Microsoft and licensed to the hardware manufacturers has been benevolent to simulation and game developers from a technical point of view. Although internally designed to conform to the uniqueness of each hardware manufacturer, it appeared consistent across brands in its command structure for use in software development. However, if developers place DOS or "system" on the disk sent to users, a paradoxical legal problem occurs.

The use of DOS at a given site is based on agreements negotiated between the DOS vendor (e.g. ITT DOS) and the user. At best, the user has a license to use rather than ownership of the product. The license applies either to the purchaser or the workstation. Therefore, software distributors or publishers must release the exercise and game diskettes without the system files and provide instructions to the user on a method of transferring DOS to the disk. However, the university environment poses unique problems to student users of published software.

Unlike the corporate user, the student usually works at "the first available workstation" and owns neither the computer nor the DOS disk associated with that PC. Since the student is not licensed, some universities have initiated policies that prohibit students from transferring DOS to their legitimately purchased software. Simulation and exercise developers who depend on the CONFIG.SYS program to specify the number of files, a versions of the BASIC programming language or the AUTOEXEC.BAT file to provide automatic loading of certain files are particularly affected by this legal technicality. The paradox thus created is an obstacle to the development and use of increasingly sophisticated simulations that the PC hardware and operating system are capable of handling.

User Support

Technical support of mainframe exercises and simulations has traditionally come from the computing center. Because of this, simulation games were distributed in source code that was modifiable by personnel trained on the specific system in place. Questions about the program and its execution were therefore directed at the on-site staff rather than the authors. The PC simulation places the user support primarily in the hands of the software developer. Thus, authors must be prepared to field technical questions (i.e. I have a _____ computer; how can I get your program to run) in addition to the more academic development questions such as how to incorporate the exercise into class curricula.

Moreover, the message of simplicity that the PC conveys has enticed computing novices into experimenting with these experiential techniques. While this is theoretically desirable, it implies that the author(s) must deliver technical service on a long-distance basis to users with a limited knowledge base.

[3] These are examples and not intended to represent a complete list.

The dilemma thus created is how to create a PC exercise that requires a minimal level of user support while providing the breadth of the mainframe activities.

Interaction with the User

A comprehensive guide to software development published in 1985 (Stahl) suggests that a user- friendly interface on PC software needed to have several characteristics:

1. the required inputs or keystrokes must be "intuitively obvious."

2. the user needed to be prevented from "entering" incorrect or inappropriate keystrokes.

One characteristic, the intuitive interface, is new to simulation and exercise developers; the other is not. Intuitive handling of the user interface requires a sophisticated knowledge of cognitive psychology. Productivity software frequently handles this issue with contextual help screens assuming that some minimal instructions will coach the user through any difficulties in use.

Another technique that has been employed to promote an ambience of intuitive behavior of the user is the use of consistent commands (e.g. press any key to continue; use ESC for the previous screen). However, the a priori knowledge of how to handle the software does not guarantee that the student or professor will "end" the program properly so that files are closed and readable for subsequent printing, iterative exercises, or transferring to instructor- readable programs. Improperly written files can ruin an otherwise extremely valuable exercise. The first user dilemma is not only how to create the intuitive keystroke pattern, but how to insure that programs are started and completed correctly.

The second issue of substance regarding the user interface is the prevention or filtering of incorrect or inappropriate responses. The most common flow of a PC-based simulation is an emulation of the batch processing environment found on a mainframe. Thus, students make their decisions on a screen version of a decision form or turn in hard copy to the instructor who then creates a decision file. An incorrect response can be filtered easily: program code defines acceptable keystrokes and prompts for a re-entry. An inappropriate response is more difficult to trap. For example, in several simulations, students may be able to overexpand capacity because the data input program is not able to filter all possible errors before processing the algorithm.

The critical point is that the user interface is created for use by non-technical instructors and students who may purposefully or inadvertently make errors that can foil the successful play of a simulation. The responsibility for addressing these problems falls on the simulation developer.

Protecting Student Files

Students are masters of deception. The experiential exercise or simulation developer ought to assume that within each student population is a potential violator of the sanctity of his or her classmate's files. The mainframe days offered some consolation since the operating system had the ability to provide some security and keep a record of user access. Though not a perfect system, it provided a reasonable probability that successful perpetrators would be detected. The personal computer does not offer the same level of inviolability.

Three major protection schemes appear to be popular methods of protecting access to files:

1. Use of a "hidden file" or other means of concealing files from the directory.
2. Use of a password system.
3. Use of internal checks that cross-reference Information indicating that the user is legitimate.

These methods interact with the previously mentioned dilemmas. The hidden file scheme requires some level of technical competence on the part of the instructor should the student disk need replacement. Password protection and internal checks are difficult to enforce when floppy disks are portable and not subject to the security of the computing center of the university.

Summary

The issues confronting the simulation and experiential exercise developer can be categorized using the standard typology for discussion of information system technology: hardware, software, people, and data. The "market" and the university environment are unique, however. Corporate software developers, whose agenda is to improve data flow for decision-making have resolved many of these dilemmas through user group and MIS department feedback. Academic developers and users, whose agenda is to synthesize learning processes (Smith and Golden, 1987; Bloom, 1955) are lacking a forum for discussing the issues associated with the PC medium.

The current act of critical issues: problems with hardware, software, users and their data needs to be addressed in the context of the university environment, instructors, and student user. This paper raises some areas of concerns and leaves them unanswered because the solutions being used at the present time are unsatisfactory. For example, some authors will make their user interface "beep" to warn of impending errors and write files on disks frequently to insure that data doesn't get misentered or lost. However, that doesn't prevent the user taking a break (answering the telephone or from walking away from a workstation) leaving the "exercise" vulnerable to variations in the electrical signal coming from the local power company. These and other concerns need to be addressed for the academic environment in the personal computing medium.

REFERENCES

[1] Bloom, B.S., Taxonomy of Educational Objectives: Handbook I, Cognitive Domain, (New York: David McKay, 1956).

[2] Burch, John, C., Strater, F. R. and Grudnitski G., Information Systems: Theory and Practice, (New York: John Wiley and Sons, 1983).

[3] Hinton, Roy C. and Smith, D. C., Stratplan, (Englewood Cliffs, NJ: Prentice-Hall, 1985).

[4] McLuhan, Marshall, "The Medium is the Message," in C. D. Mortensen, ed. Basic Readings in Communication Theory, (New York: Harper Row, 1979).

[5] Schreier, James and Dondliner, T., Management for Productivity: An Electronic In-Basket, (New York: John Wiley and Sons, 1986).

[6] Scott, Timothy and Strickland, A Micromatic (Boston: Houghton-Mifflin, 1985).

[7] Smith, Jerald and Golden, P. Airline: A Strategic Management Simulation, (Englewood Cliffs, NJ: Prentice-Hall, 1987).

[8] Smith, Jerald and Golden, P. "Reaching for the Affective Domain: The Management Development Jungle," Journal of Management Development forthcoming 1987.

[91 Stahl, Bob, "Friendly Mainframe Software Guides Users Toward Productivity," Computer World, February 3, 1986, pp. 53-66.