PAINLESS COMPUTER SIMULATION

Terry L. Dennis, Rochester Institute of Technology

ABSTRACT

Using a computer simulation in class does not have to be a painful experience. This paper discusses recent approaches to simulation programming which: make student data entry interactive; automatically check for and even supply missing data; and use command files to combine numerous files and run the simulation with minimal administrative effort.

INTRODUCTION

Painless Computer Simulation

In the past, using a computer simulation as a pedagogical tool frequently turned into a painful experience for the faculty member. New versions of computer languages, time sharing capabilities, and greater sensitivity on the part of the simulation programmer can change that experience to a more positive one.

Historical Perspective

The "simulation administrator" (also known as the professor) has just finished key punching the last of the administrator's cards, being careful to right justify each entry in the exact location specified. He then collects the student decision cards from each student and carefully sorts them into the proper order. Then the control cards, administrator cards, decision cards and history cards are assembled together and this deck is carried to the computer center and submitted. Anywhere from several hours to several days later, the administrator returns to collect his input deck, output deck, and computer printout. At this time, he discovers one of the following outcomes:

(1) a mispunched card caused an error and the program aborted -- find the mistake and try again.
(2) a card out of order caused an error and the program aborted.
(3) the program ran properly -- but a mispunched card caused the output to be meaningless for one or more participants, meaning either a rerun of the program for the entire class or the only learning experience for those students was in key punching.
(4) the program ran properly and students could relate the course material to their simulation results.

Hopefully, the result was the latter one and the administrator could begin again for the next period of play. If the above scenario brought back vivid memories (nightmares) you are not alone. Many instructors gave up on simulations because of the hours of extra time required and the headaches involved.

Fortunately, those problems are becoming things of the past for most faculty. Time sharing terminal input and data files have solved the problem of handling large decks of cards. (although some thought the boxes and stacks of cards piled around the office were impressive to visitors). More compassionate versions of FORTRAN[1] allow unformatted entry of data and opportunities to catch mistakes which once would have terminated programs prematurely.

[1] For example, American National Standard FORTRAN 77 (ANSI X3.9 - 1978).

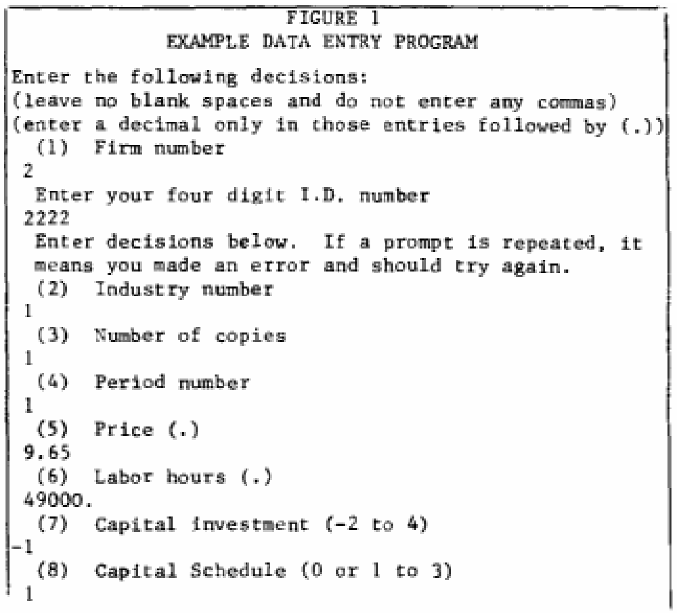
And the authors of simulations are beginning to take advantage of these features to further ease the burden on both administrators and students.

A recent article[2] summarizes a number of desirable points for simulations which would make them easier to administer. This discussion paper is meant to encourage simulation authors to be responsive to those points whenever possible. By creating "user friendly" input programs which can be used on time sharing systems in conjunction with command programs, authors can make their simulations easier to use and, therefore, better learning experiences for students. A data entry program and command program created for the microeconomic simulation MICROSIM[3] will be used as examples

Data Entry

Since many problems occur with data input by students, a separate time-sharing program named ENTER was created to insure correct, or at least reasonably correct, student input. Once students make their decisions, they go to a terminal or CRT and run the ENTER program (e.g., RUN ENTER). The program begins with a brief set of instructions and a prompt for the firm number. When the student enters the firm number, a four digit identification number is requested for security purposes (this prevents unauthorized users or unscrupulous competitors from entering erroneous or bad decisions). If the student enters the correct ID number, the program proceeds with prompts for the remaining decisions as shown in FIGURE 1.

FIGURE 1
EXAMPLE DATA ENTRY PROGRAM



```
Enter the following decisions:
(leave no blank spaces and do not enter any commas)
(enter a decimal only in those entries followed by (.))
  (1)  Firm number
 2
  Enter your four digit I.D. number
 2222
  Enter decisions below.  If a prompt is repeated, it
  means you made an error and should try again.
  (2)  Industry number
 1
  (3)  Number of copies
 1
  (4)  Period number
 1
  (5)  Price (.)
 9.65
  (6)  Labor hours (.)
 49000.
  (7)  Capital investment (-2 to 4)
-1
  (8)  Capital Schedule (0 or 1 to 3)
 1
```

[2] Robert H. Dunikoski and Richard F. Barton, "Super Service for Computer Game Administrators, ABSEL News & Views, Vol. 2, No. 2, p. 6,

[3] Steven C. Gold, Thomas F. Pray, and Terry L. Dennis, MICROSIM: A Microeconomic Simulation, Macmillan Publishing Company, c. 1983.

FIGURE 1 (Continued)

```
  (9) Advertising budget (.)
 150000.
 (10) Development budget (.)
75000.
 (11) Economic information (0 to 7)
7
 (12) Market information (0 to 19)
19
```

If the wrong ID is entered, the student is given another chance. If the correct ID is not entered after three times, the program terminates itself (FIGURE 2).

FIGURE 2

```
 (1)  Firm number
2
 Enter your four digit I.D. number
2323
 Wrong I.D.  Try again
 Enter your four digit I.D. number
2324
 Wrong I.D.  Try again
 Enter your four digit I.D. number
2325
 I am giving up on this.  Come back when you have the
 correct I.D.
```

Once all decisions have been entered, the entries are summarized and the student is asked if he or she made any mistakes. If the response is negative, the entries are automatically added to a data file for student decisions by the computer, properly formatted for use by the simulation program (FIGURE 3)

FIGURE 3
INPUT SUMMARY

```
 You have input the following decisions:

  (1)  Firm number:              2
  (2)  Industry number:          1
  (3)  Number of copies          1
  (4)  Period number:            1
  (5)  Price:                    9.65
  (6)  Labor Hours:          49000.
  (7)  Capital investment:      -1
  (8)  Capital schedule:         1
  (9)  Advertising:         150000.
 (10)  Development:          75000.
 (11)  Economic information:     7
 (12)  Market information:      19

 Did you make any mistakes?  (Y or N)
N

 Your data has been entered
```

But what happens when the student makes an entry error anyway and falls to correct it when given the opportunity? Further protections are built into the program. If the student enters a decision which is blatantly in error, e.g., the number 500 entered with the digit five followed by two letter 018, the program reprompts for the same decision. If the student enters 9's for economic information despite being prompted to enter a number between 0 and 7, the entry Is not accepted and the prompt is repeated. Since students occasionally enter commas in large numbers despite being warned not to (a comma is interpreted as the end of an entry,' thus 50000 is read as fifty-thousand while 50,000 is read as fifty), the program cannot distinguish between a correct entry of fifty and an incorrect entry of 50,000. While it must allow a real decision of 50, if the decision is an unlikely one, the program Issues a caution (see FIGURE 4) and reprompts for verification.

FIGURE 4
ERROR CHECKING ROUTINS

```
 (6)  Labor hours (.)
50,000
 You have scheduled less than 1000 hours.
 You may have included a comma (e.g., 50,000) by
 mistake.
 Please re-enter your decision as a check.
 (6) Labor hours (.)
50000.
```

If a 50 is entered a second time, it is accepted and the program proceeds. What happens if the student forgets a decimal? One is added to the end of the figure entered, e.g., 6000 becomes 6000. While this program is not fool-proof -- It cannot tell if the student meant to enter 7000 instead of 6000 and didn't catch the mistake in the summary, nor will it correct plain bad judgement, e.g., a price set below cost -- ft has eliminated the most frequent errors made by students, especially those which cause unexpected termination of the simulation.

Two additional problems sometimes encountered, decisions out of order and missing decisions, are handled by the simulation program itself. If decisions for a particular firm are missing, the program automatically supplies them. The decisions supplied are not particularly good ones, but neither are they so bad as to make later recovery in an ongoing simulation impossible, which could reduce involvement and motivation. Decisions not in order by firm number cause no problems; the simulation orders all decisions internally. The period number is also indexed automatically, making it unnecessary for the administrator to change a value each period (beyond period 1).

Running The Simulation

Once the administrator has established the necessary industry and environment parameters, he may use a special command file to run the simulation for additional periods, by issuing one instruction to the computer via a remote terminal, e.g., RUN MS, the administrator allows the computer to process a set of pre-established commands which merge the administrative file, the student decisions, and any history into one input file, run the simulation, create a new history file, and send output to the printer. At this point, there are two options: (1) send all output to a printer where someone must pick it up and distribute it to the class; or (2) have each firm's output sent to a file in the students' individual accounts so they can access them, as they desire. Currently the first option is being used because it guarantees that all firms will have their output for reference during class discussions.