CHAPTER 5

THE EDUCATION OF AN ENVIRONMENT DESIGNER

Importance of a Certain Minimum of Mathematics and Computer Science for Environment Design

This will undoubtedly be the most controversial chapter in the book because it argues that a good Environment designer must have a certain minimum knowledge of basic programming concepts — must be a citizen of the technical world in which he or she earns a living and, in fact, lives. Let me immediately make clear to my potential critics that I am not talking about the requirements for a career in technical writing here: many people earn a good living in that field knowing little about any technical subject.

Programming supplies the *templates* for thinking about Environment design and implementation. Let me give an elementary example: a technical writer was having a difficult time explaining a calibration procedure for an electronic test instrument. She invited several of her co-workers, none of whom had any programming knowledge or experience, to offer their opinions. The discussion went on for several days. But the *form* of the procedure she was trying to write was immediately clear to anyone who knew what a *while* loop is in programming. In fact, the form was simply this:

while
<(value of parameter p_1 is outside the range r_1 or
value of parameter p_2 is outside of range r_2 or
.
.
.
value of parameter p_n is outside of range r_n) and
number of times through this loop is less than 5>
do
begin

Make adjustments as follows...

end

But the war of words continued even after this simple solution to the problem was pointed out. (The audience for the manual, incidentally, was programmers.)

The best illustration of the universal applicability of programming concepts that I have come across is an essay that arrived from a friend via one of the computer nets one day several years ago.

"Mother's Day Meditations from the Computer Room

by Gertrude Martin's son David (martindm@acm.org)

"A friend recently asked me what training it takes to work with computers. I gave a brief answer mentioning some college courses, some on-the-job training, and a long time in the school of hard knocks. But upon reflection, I realize that most of my training in fundamental computer concepts came from my mother.

"When I was a baby, mother taught me about input buffering: 'Don't try to stuff all your food in your mouth at once. Leave it on your plate until you're ready to eat it, and then take it in one mouthful at a time.'

"She also taught me about processing the entire input buffer before going on to the next step: 'Eat everything on your plate. Then you can have dessert.'

"(It will occur to some readers that mother also taught me about output buffering, but I'd like to keep these meditations G-rated.)

"When I was about four, mother introduced the concept of sequentially executed instructions: 'We're going to set the table.' (That's identification of the procedure.) 'First put the table cloth on the table. Check it to make sure it's straight. Then put a plate at each place. Then put a cup at each place. Then...'

"Later, mother introduced the concept of a procedure call: 'We're going to have dinner. Please set the table.'

"Still later, when I was about 14, mother would set up tasks for me and use 'job control language' in a note on the refrigerator door: 'We're going to have dinner at 6:00. You make it when you get home from school. The menu is pinned up on the bulletin board, the meat is in the refrigerator, and I've put the rest of the food out on the counter. Set an extra place — Uncle Jack is coming tonight.'

"Mother demonstrated what it means to multi-process: She could deal with the interruptions of four children (those were the real-time, foreground tasks) while doing the housework (as a background task).

"Mother used the concept of hierarchical storage for her cooking tools. The cooking forks and spoons were hung on hooks right by the stove. The potato slicer and the egg beater, which weren't used for every meal, were kept in a drawer. And the

big roaster, which she only used once a year to cook the Thanksgiving turkey, was kept in the storage closet in the basement.

"Once we had about fourteen people for Thanksgiving dinner, and our kitchen seemed too small for the job. That's when mother introduced the concept of backing store. She cleared off the ping-pong table in the rec room next to the kitchen and laid out all her ingredients on one side of the net. My sister and I fetched things from the 'input' side of the ping-pong table as mother called for them, carried partially finished dishes to and from the 'backing store' on the other side of the net, and delivered finished food to the 'output' dining table.

"This system worked well, until my sister and I collided in the doorway between the two rooms and we nearly lost the creamed onions. Mother solved this problem of 'channel contention' by establishing a protocol: 'First say "May I come through?" and then wait until you get the answer "Yes, it's clear."'

"It was also in the kitchen that mother taught me about looping and testing: 'Cook the fudge, while stirring it, and test it every couple of minutes to see if it's done. You test it by dropping a bit of it in the cold water. When it forms a soft ball, it's done.'

"For years I badgered my mother with questions about whether Santa Claus is a real person or not. Her answer was always 'Well, you asked for the presents and they came, didn't they?' I finally understood the full meaning of her reply when I heard the definition of a virtual device: 'A software or hardware entity which responds to commands in a manner indistinguishable from the real device.' Mother was telling me that Santa Claus is a virtual person (simulated by loving parents) who responds to requests from children in a manner indistinguishable from the real saint.

"Mother also taught the IF...THEN...ELSE structure: 'If it's snowing, then put your boots on before you go to school; otherwise just wear your shoes.'

"Mother explained the difference between batch and transaction processing: 'We'll wash the white clothes when we get enough of them to make a load, but we'll wash these socks out right now by hand because you'll need them this afternoon.'

"Mother taught me about linked lists. Once, for a birthday party, she laid out a treasure hunt of ten hidden clues, with each clue telling where to find the next one and the last one leading to the treasure. She then gave us the first clue.

"Mother understood about parity errors. When she counted socks after doing the laundry, she expected to find an even number and groaned when only one sock of a pair emerged from the washing machine. Later she applied the principles of redundancy engineering to this problem by buying our socks three identical pairs at a time. This greatly increased the odds of being able to come up with at least one matching pair.

"Mother had all of us children write our Christmas thank you notes to Grandmother, one after another, on a single large sheet of paper which was then mailed in a single envelope with a single stamp. This was obviously an instance of blocking records in order to save money by reducing the number of physical I/O operations.

"Mother used flags to help her manage the housework. Whenever she turned on the stove, she put a pot holder on top of her purse to remind herself to turn it off again before leaving the house.

"Mother knew about the devices which raise an interrupt signal to be serviced when they have completed any operation. She had a whistling teakettle.

"Mother understood about LIFO ordering. In my lunch bag she put the dessert on the bottom, the sandwich in the middle, and the napkin on top so that things would come out in the right order at lunchtime.

"There is an old story that God knew He couldn't be physically present everywhere at once, to show His love for His people, and so He created mothers. That is the difference between centralized and distributed processing. As any kid who's ever misbehaved at a neighbor's house finds out, all the mothers in the neighborhood talk to each other. That's a local area network of distributed processors that can't be be beat."

Programming Concepts

Here is a list of basic programming concepts that an Environment designer should understand. They are very simple, as you can see from the previous section. The designer can get brief explanations on Google or by browsing in the Computers/ Programming section of any large book store, or by talking to any programmer.

- variables
- loops
  - *for* loops
  - *while* loops
  - *do-until* loops
- recursion
- data types
- functions
- subroutines, procedures (these, and functions, make possible the structuring of programs).

Getting Along with Engineers, Programmers, Managers

Some points that every Environment designer must keep in mind about engineers, programmers, and managers:

- they don't see the world as you do: specifically, they see technology, products, programs, hardware as central; you see the *use* of these as central.
- your profession is a necessary evil to them; they wish this whole, annoying, problem of usability would go away, so that they can concentrate on the important things;
- the more of their kind of technical knowledge you have, the greater the respect you will get from them.

Selling New Ideas to the Same

In selling new ideas to engineers, programmers, and managers, talk profit, talk beating the competition.

Jobcraft

There is a set of good behaviors you can apply. See Appendix D for details.