A Cautious Hurrah

Elihu M. Gerson

My last few columns have touched upon a variety of topics which are seemingly unrelated: the value of acquiring programming skills, the need to frame our data in terms of structured "objects," the value and limitations of hypertext, the importance of "smart environments" which know something about what we are likely to want. There is a common concern underneath these columns; they all discuss aspects of current "frontier" computing technology which will be an important part of truly serviceable technology.

Until now, there has been no commercially available software which incorporates all of these features at the same time. Hence, it has been impossible to provide a clear example of the kinds of services such a smart, civilized technology might perform for us. Indeed, only one experimental system, the Notecards system developed by Randy Trigg at Xerox Palo Alto Research Center approximates the kinds of facilities we need (Trigg & Irish, 1987; Morell & Trigg, 1987). Recently however, Apple Computer has released a program called Hypercard, which is now being bundled with all new Macintoshes sold. Hypercard incorporates much of the computer science thinking which I have been drawing on in these columns. There has already been so much publicity about the program that you are certain to be familiar with it if you own a Macintosh. I don't, and so my acquaintance with it is limited to a few demonstrations and the independently published documentation (Goodman, 1987). So this is not a review of Hypercard; rather, it is a way of pulling together the points I have been making, using Hypercard as a working example.

With that disclaimer in mind, I should say at the start that the first demonstration of Hypercard I saw convinced me on the spot to abandon my IBM PC and switch to the Macintosh, and it has only been the lack of competent word processing software and Apple's truly outrageous
prices which have kept me from doing so. But no doubt all that will change, and I’ll switch very soon now. Metaphorically, Hypercard consists of a set of “stacks,” which are analogous to decks of 3 by 5 index cards. Each card can contain a number of “fields,” which allow you to organize data in various ways. Cards may also have graphics, and the program has an excellent “paint” facility built in. Finally, there are “buttons,” which allow you to trigger a sequence of actions by “clicking” at a designated point on the card image. Clicking on a button signals Hypercard to run whatever program has previously been associated with the selected button. Programs can be arbitrarily complex, triggering anything from simply moving to the next card in the sequence, to an interactive session in which new programs are specified, created, and executed. Buttons can be associated with individual cards, with types of cards, with stacks, or with the system as a whole. These facilities provide Hypercard with all the advantages of a true, if limited, object-oriented programming language.

Hypercard is not the answer to all our problems; far from it. The programming language (“Hypertalk”) is missing some important features, such as arrays and a case statement. The file import/export facilities are confined to simple ASCII, which can be organized by the read/write facilities of the program. This is very weak, and needs much improvement. There is no “navigational” facility, so that you cannot see an overview of where you are in a complex network of connections. Nor, of course, can you record the state of that network at a point in time and manipulate it later as data. Using the program to retrieve sets of records, the way you would a database manager, is apt to be painfully slow, although single-record retrieval is quite fast. You can only have one “card” (i.e., record or form) on the screen at a time. The report writing facilities are extremely limited. And so on and on, through a long list of limitations.

Despite its faults and limits, Hypercard can easily be made to handle all the basic tasks of entering field notes, summarizing them, coding according to analytical categories (Strauss, 1987), sorting and retrieving passages by category, and so on. The programming language (which is quite simple to learn and use) allows setting up very complex analyses fairly easily. It thus has all the features I envisioned in my paper on “Qualitative sociology and the computer” (Gerson, 1984) and a good bit more besides. In that sense, it is certainly one of the most useful programs currently available for personal computers. Moreover, it is clear that many different support facilities for qualitative research with Hypercard will be available very soon via the usual informal academic channels, if not formal publication.

But the possibilities and promise of this class of programs extend far