# THE GRAPHICS SOFTWARE FAMILY

Mark G. Rawlins

Marketing Manager
ISSCO Graphics
San Diego, CA

## ABSTRACT

Graphics is widely accepted as a valuable tool for transforming data into information. Computer graphics is becoming recognized as the perfect medium for producing charts quickly, economically and accurately without sacrificing aesthetic quality. Today's DP manager is now perplexed, for he is now faced with new responsibilities:

1. He must introduce computer graphics in his organization.
2. He must be concerned with machine portability and device interfacing.
3. He must evaluate the capabilities and support of available packages.
4. He must be aware of who the end-users will be and what sorts of applications they'll have.
5. He must understand how graphics software can be tied to existing application programs or how data files can be accessed.

There is no one package that can satisfy all the needs that will arise. A family of graphics software products, however, can address all these concerns.

Family members have common lineage but unique personalities. A family of software packages developed by a single source has built-in consistency at the system end and versatility at the user end. The DP manager's problems are solved and the needs of the end-users are easily satisfied.

KEYWORDS: graphics software, application programs, portability, productivity

Estimates have been made that software will comprise 13% of the Gross National Product of the United States by the year 1990. Corporations are turning to software as a backbone to their productivity tools. In the last decade, computer costs have dropped so significantly that anyone can now afford a computer. The only question surrounding the ultimate use of computers is "how to make the software available in an environment where it will be used".

Application programs have been around since the days of the first vacuum tube computer. Getting these programs to be used by people other than the programmer has created the environment where the "medicine is worse than the disease". All of the computing power in the world will not make a manager use software tools if the "grief factor" is too high. The application software must be a logical extension of the user's mental process. "If I can use the computer as easily as I use my pencil, then I will use your software", said one middle manager to the director of data processing of a major east coast bank. Another often heard statement is, "When I ask for more information all I get is more printout. I don't want more data - I want more information".

ICP, a major directory listing of commercially available software packages presents but a snapshot of application software in about 400 pages. This listing does not include software written in-house. The total number of application software systems is staggering. The concern now arises in how an organization makes these (or any) software tools more usable. How can they transform the data from the computer into information for the user. This is a very broad problem that has no single solution. Among the concerns when analyzing this problem are: education and training of the user, user friendliness of the software, up/down ratio time of the computer, ease of access to the data, etc, etc. A proper match needs to be made between the sophistication of the user and application. Does the software need to be written for the novice or the expert? How "user friendly" does the program need to be? Each piece of software needs to be viewed not only by itself, but also in its relationship to other software within an organization. The programs must be evaluated on their individual merit: who uses them, how they are used, supporting resources for the software, and what other ancillary software is necessary to make this programming environment truly effective as a family of productivity tools.

The statement which was made before, "I don't want more data-I want more information", reflects a concern which can affect any software, whether it is commercially available or written in-house. Programs gather or create data, analyze it and then present it to the user. This path of information is actually many paths. Every application might have its own method of taking commands from the user, processing the data, and offering it for presentation. These paths might be further complicated when the environment runs the spectrum from highly sophisticated to ultra simple. It is in the area of "presentation" that graphics has its greatest strength. Imagine, if you will, a data base retrieval system in which information is being correlated in order to show trends. Few will deny that the best way to present this trend information would be graphically. The end user would certainly find more value and productivity in his application program if graphics would also be produced. Of comparable importance now is the ease of production.

The computing environment 10 years ago made graphics a wish, not a reality. Happily, the physical problem of getting graphics out of the computer no longer exists. That excuse is fortunately gone forever. Technology exists today to solve virtually any hardware problem. A new concern now arises: how do we get graphics out of the software--sophisticated graphics for the expert and easy-to-use graphics for the novice?

People swear that properly getting graphics and data together for presentation is not that easy. The data processing department has to be concerned about consistency of graphic quality across applications, maintenance, flexibility, hardware considerations, training of both support people and users, cost, ability to run interactively or in batch, and will the software require operating system modifications. This goes further than just a programming concern. Once the software is written, will it be used? There is an obvious obligation to the organization to supply the most productive tools possible.

Just as General Dynamics supplies the fuselage for the DC-10 which McDonnell Douglas builds, the data processing department has at its disposal software houses (independent contractors if you will) which specialize in doing just graphics. The above-mentioned problems can now become more manageable.

In looking just at the hardware considerations, some graphic software does not have the proper device independent versatility. In looking at

the cost factor, it may be that operating system modifications need to be performed. In looking at flexibility, it might be that the graphic software can only run in batch. In looking at ease-of-use, a programmer might always be required.

What's necessary is a family of graphic software which has the same graphic consistency among its products: the same maintenance environment, a high value/cost ratio, the ability to operate all graphic devices on any computer type, and an acceptable level of sophistication and quality. The graphics supplier should also help with maintenance, installation and system support, as well as training.

Once the support questions have been answered, the data processing department has taken care of only the tip of the iceberg. What about the needs of the user? Will they want bar charts, line charts or pie charts; 3-D or mapping; contouring or typesetting? What about sophistication, ease of input, type of output?

In breaking down the application software market into categories where source is supplied and those in which there is no source, the industry has created a requirement for a graphics option to exist in both programming form and in stand-alone turnkey form. For the software packages where source is available, it is apropos to imbed references or calls to graphic subroutines. Applications which do not have source need to have an easy mechanism to pass data directly to a graphic package.

Analyzing the user community, potential users of graphics software fall into two categories: sophisticated programmers and non-computer professionals requiring graphics tools. Programmers are adept at manipulating graphic subroutines to achieve results. Novice computer users are at the mercy of easy-to-use turnkey systems to help them in their work.

Both of these graphic environments must offer consistency in quality, flexibility, and device support. The end-user is not concerned with the internals of the package he is using. His concern is that the plot drawn by Application A is consistent with the plot drawn by Application B. The user also wants the same high quality whether running interactively or in batch. All these expectations (and more) must be met, easily.

It is now obvious that the supplier of graphic software must offer more than one product. If the application package has source, a programmer could imbed routines so charts automatically result at the report-generation stage. If there is no source or if the end-user lacks programming expertise, then the data must be accessible by the graphic stand-alone product.

In putting the graphics into the program, we accomplish many things. The graphics can be optimized around the application or the data. By having access to a graphic subroutine library, just the graphic entities which are necessary will be brought into the application program. For example, if the user has no need for generating pie charts, then reference to the programs that produce pies can be omitted. This obviously cuts down on memory usage and thus minimizes resource impact.

By imbedding graphic subroutine commands in an application package, the command structure to which the user has become familiar can be maintained. It has always been known that a stumbling block in the use of software tools is the "fear of use". If management has been trained to use existing tools, the last thing needed is to make the tools more difficult or convoluted. Enhancing a system by incorporating graphics is not meant to scare or frighten users away. It is very important to maintain command friendliness and structure the graphic commands to be very similar to the trusted syntax. It is also necessary to maintain the confidence in proven software. Old procedures are always used with more confidence and trust than new procedures. In general, "it is never easy enough to use a program which increases productivity".

The second environment of directly accessing data by a graphics stand-alone package can create many more concerns, especially if the proper graphics software is not chosen. Remembering that this area is reserved for non-computer end-users and/or application software which does not have source available, a linkage must be created to couple data to the graphics product. This linkage can be via the data generated by the original package.

Normally, application data is either generated in a rectangular binary file (work file) or in a character-oriented file which can be printed on the system line printer. As data generation occurs naturally in both forms, the user should be able to take advantage of each technique depending upon the application. The graphic portion should be able to automatically

latch onto the machine readable file or the print file. Once the data has been created, the graphic program should use simple, English-like commands to read the data, regardless of format, extract some data, if necessary, and plot it. This two-step process maintains the integrity of the original package and maintains the user's confidence in his software tool. The standard output technique can be made transparent to the user and thus non-threatening. The user also gets graphics when desired. A tall order, but quite fillable.

The variety of application, the diversity of end-users and the DP manager's requirement for some sort of consistency justifies the need for a family of graphics packages. As graphic output is a necessary tool for adding value to data, data access is also a crucial problem about which the DP manager must be concerned. By offering users the ultimate in productivity tools, a graphics software family is the only technique available to keep both the data processing department and its end-users happy.