CPG MEMO NO. 27

National Radio Astronomy Observatory
P.O. Box O
Socorro, New Mexico 87801

MEMORANDUM

To:	Computer Planning Group (CPG)
From:	Jim Torson
Date:	December 15, 1983
Subject:	Recent Graphics Standardization Discussions

A memo by Don Wells [1] discusses the proposal to adopt a standard graphics package at NRAO. The following are some miscellaneous additional comments concerning this proposal.

1. CRITEREA FOR CHOOSING A GRAPHICS PACKAGE

Other institutions want to obtain applications software produced at NRAO and run it on their own computers. Obviously they would like to do this without paying a licensing fee for the graphics package that is used by our software. One way for this to be accomplished would be for us to adopt a graphics package that is in the public domain. However, there may be other ways. If we decided to use a commercially available graphics package, we might be able to make an arrangement with the vendor whereby NRAO would make an appropriate payment and then the various interested institutions would be able to use the package with our software without paying further fees. To my knowledge, nobody with the proper authority at NRAO has seriously investigated this possibility with any of the vendors.

Perhaps we should reconsider the requirement that other institutions must be able to run our software without paying a licensing fee for the graphics package. If an institution wants to use our software for data processing, they must obtain some hardware tools, i.e., an appropriate computer. They must also obtain some software tools, i.e., a Fortran compiler. If they want to use our software for graphical data display, they must obtain some hardware tools, i.e., some sort of display device. Perhaps it would also make sense to require them to obtain some software tools, i.e. a graphics package.

2. GRAPHICAL KERNEL SYSTEM AND THE SIGGRAPH CORE SYSTEM

Reference [2] gives a brief description of the Graphical Kernal System (GKS) and a summary of the history of graphics standardization efforts. The Core graphics system was not developed by ANSI. Instead, it was proposed by the ACM SIGGRAPH Graphics Standards Planning Committee (GSPC) in 1977 [3]. A revised proposal was published in 1979 [4]. At this time the GSPC ceased to exist and an ANSI subcommittee was created to continue the standardization efforts. In any event, as a result of this work plus other work in Europe, we now have GKS as a draft proposed International Standard and a draft proposed American National Standard. The latest version of GKS is described in [5].

It is important to keep in mind that GKS (and Core) is a standard for the interface between an application program and the graphics system. It specifically does not attempt to specify the internal details of how the graphics system is implemented.

It is not true that GKS is expected to evolve to include anything which is now missing from it. GKS is not designed to fully handle the dynamic high performance displays. Instead, a separate standard called Programmer's Hierarchical Interactive Graphics Standard (PHIGS) is being developed for these types of devices [6].

Although there is much similarity between GKS and Core, an important difference is that GKS does not support 3D functions. Of course this problem is being worked on, but some members of the graphics community have suggested that GKS is not a suitable standard on which to base three dimensional extensions [7]. The SIGGRAPH Executive Committee has voted for a plan to adopt the Core System as the first ACM standard, and then to submit it through the ACM Standards Committee for processing as an American National Standard. "The SIGGRAPH Executive Committee took this action for two reasons. First, there are a large number of users of the Core System. The Core System has become a de facto standard, and this action will legitimize it as a standard. Second, there is a need for a 3D computer graphics standard. The development of such a 3D standard appears to be several years off, both within the U.S. and the international standards making bodies. By SIGGRAPH taking this action now, those users who need a 3D graphics standard will have one at a much earlier date." [8]

3. VIRTUAL DEVICE INTERFACE AND VIRTUAL DEVICE METAFILE

Several separate ANSI/ISO standards relating to graphics are being developed. GKS is the specification of the interface between applications programs and the graphics system. Work on this standard is nearing completion.

A second standard being developed is the Virtual Device Metafile (VDM). The VDM is a specification for a device-independent data format for computer graphics pictures. The purpose of the VDM is to provide a means of device-independent picture transfer, either between different graphics devices, different computers, or different graphics packages. Work on the VDM standard is nearing completion. Reference [9] describes the latest version of VDM. The VDM would typically consist of a disk or tape file. The standard includes a specification for both a character encoding and a binary encoding of the individual elements of the VDM. It does not specify how this information is transmitted or recorded. The VDM deals only with graphics output.

A third standard being developed is the Virtual Device Interface (VDI). The VDI is a specification of the interface between the device-independent and device-dependent parts of a graphics package. Reference [9] contains a description of the relationship between the VDM and the VDI. It also describes the basic VDI model and the philosophy of how the VDI standard should work. However, detailed specification of the VDI standard has just begun. The VDI will be partitioned into a lean set of required functions and a rich set of nonrequired functions. The VDI will include input functions as well as output functions.

4. DEVICE DRIVERS

The device-dependent part of a graphics package is called a device driver. It translates the device-independent information into the form required by the physical device. Of course a separate device driver is needed for each different type of device that is to be used.

It has been suggested that hardware vendors will supply device drivers to go with their hardware devices and that one vendor is already working on a driver for its device. There are several reasons why I doubt that this is the case. First, since the VDI functions are not yet defined, any GKS implementor must define his own version of the VDI. A vendor's implementation of a device driver would probably not match all of the different VDI's that are being used by the various different GKS implementations. Second, GKS does not specify implementation details such as how information is communicated between the device-independent and device-dependent portions of the graphics package. Also, I doubt that VDI will specify these types of details. Thus, in one GKS implementation the information may be passed as a list of parameters in a subroutine call. In another, it may be passed as a list of values packed into a single array. In yet another, the device driver may be a separate task with the VDI information passed with an inter-task communications facility. A third problem is that a device driver implementation may want to make use of lower-level utility routines that are used by more than one driver. Of course each GKS implementation will probably have a somewhat different set of such routines. A fourth problem concerns the nonrequired functions in the VDI. The implementor decides which nonrequired functions are to be supported. Also, he decides whether their emulation will be provided by the device-independent software, in the device driver, or at the device itself. Such decisions need to be coordinated between the device-independent implementation and the device driver implementation.

There is a possibility that a hardware vendor will provide a stand-alone program that will read the VDM file and output the picture on his device. However, this also seems unlikely since the GKS concept of a metafile is that the GKS code itself is used for interpreting the metafile. Thus, the hardware vendor would have to implement some (most?) of the device-independent part of GKS in addition to the device driver for his particular device. Even if he did this, it would not provide any interactive capability, so in many cases it would hardly be a strong selling point for his device.

Yet another possibility would be for a vendor to build a device which directly processes the VDI functions (when they get defined). Presumably the device would accept at least the required functions so the software device driver would just pass these on to the device. However, you would still have the question of what to do about the nonrequired functions. When (and if) vendors will build VDI devices remains to be seen. Some people have suggested that we should not expect to see hardware vendors producing VDI devices which are all compatible with each other since a prospective customer would then have less reason for buying a particular device. Instead, vendors will continue to provide devices which have special features which they think will induce customers to choose their device rather than a competing device.

On the other hand, a hardware vendor may in fact decide that a certain GKS implementation is popular enough to justify the effort of producing a device driver that will be compatible with that particular GKS implementation.

The actual functions in the virtual device interface that are chosen by an implementation can have several implications [15]. At one extreme, a small set of simple functions can be chosen. In this case the device drivers are relatively easy to develop since they only need to understand a limited command vocabulary. More complex functions are emulated in the device-independent code by breaking down each function into an appropriate sequence of the simple functions supported in the VDI. A disadvantage of this approach is that it makes it difficult or impossible to use the more powerful features of the newer intelligent terminals that are becoming available. Only the hardware capabilities corresponding to the functions in the VDI can be used.

At the other extreme, a large set of functions can be chosen for the VDI. This of course means that the device drivers will be more complex. However, this makes it possible to utilize the more powerful features of the newer terminals. For example, if a terminal has the capability to draw dashed lines, then this capability can be used. A single command to the terminal will draw a dashed line. If the VDI did not include dashed lines, then a dashed line would have to be emulated by drawing multiple small line segments. The use of a large set of functions in the VDI can thus decrease the communications between the host and the graphics terminal. Also, this approach can more easily accommodate a network or distributed processing environment. The device driver can be moved into a separate computer or even into an intelligent programmable workstation. This would reduce both the memory and CPU requirements in the host computer.

5. SUPPORT FOR IMAGING FUNCTIONS

It has been suggested that GKS contains good support for imaging functions. It is true that GKS (and the Core System) provides support for raster graphics. However, this is not the same as imaging, and I doubt that the functions provided are adequate for our imaging needs. For example, GKS contains an output primitive called CELL ARRAY which allows the specification of a rectangular array of values which are indices into the color look-up table. This of course could be used to load the image memory. However, the cells are points in world coordinates and full transformations and clipping are to be applied to each point. Loading would thus be very inefficient. What we really want is a way to load lines of the image memory as quickly as possible. Another problem is that GKS only allows a single color look-up table for each workstation, but a typical image display device will have several or many look-up tables. Also, GKS defines a workstation as containing only a single display surface, but a typical image display will have several image planes. Perhaps different image planes could be defined as separate workstations, but it's then not clear how you could handle such things as split screen, mosaicking, true-color display, or intensity/hue display. Although it may be possible to control all of the typical imaging functions through standard GKS functions, I think it would be awkward at best. You would really be using GKS to do things that it was not designed to do. It would certainly be nice if a device-independent imaging package existed, but I have not seen any indication that GKS will evolve in that direction.

Both GKS and the Core system allow "escape" functions, which are a "standard way of being non-standard." An escape function allows you to directly access a special feature of a particular hardware device. An imaging device could be controlled by escape functions, but if most of the control of a device is done through escape functions, then you have lost the advantage of using a device-independent graphics package.

6. THE NCAR GRAPHICS PACKAGE

The NCAR graphics package [10] consists of two basic parts: the System Plot Package and the Graphics Utilities. The System Plot Package provides the application-independent primitive operations. The functions it provides are mostly much simpler than either GKS or the Core System. On the other hand, it also includes some application-specific functions such as drawing labelled axes. Only 2D plotting is supported by the System Plot Package. In the standard exportable package, a device-independent metacode description of the picture is written to a file and a separate program reads the file and does the actual output to a specific device. This of course does not allow interactive graphics. It has been said that interactive capability is available in the VAX version, but this is not described in the standard documentation package. It would be useful to find out more about how this works and how easy it would be to export it to different computers. It would also be useful to find out which graphics devices are supported by NCAR. The documentation mentions only a Dicomed, but doesn't specify which model.

The NCAR Graphics Utilities consists of a package of routines for doing various applications-oriented data plotting. This includes such things as fancy contour plots, 3D mesh surface plots, and 3D iso-valued surface plots. These routines call the routines in the System Plot Package to do the actual work. Since the functions in the Graphics Utilities are oriented toward a specific application, they are beyond the scope of the functions contained in the general-purpose GKS or Core System packages.

If my understanding is correct, NCAR plans to implement a version of GKS. They will then provide a package of interface routines which will look the same as the System Plot Package to the caller but which will internally make calls to the new GKS functions to do the actual work. This will allow the existing Graphics Utilities to continue to be used without change.

The initial implementation will conform to level "0a" of the GKS specification. Input level "a" means that no input functions will be provided. Output level "0" means that only a single workstation will be available at a time. Thus, handling multiple image planes in an image display might be a problem (if you try to use GKS to handle the image display). Level "0" does not require the metafile capability. The GKS specification allows an implementation at a given level to include functions of a higher level. Since use of a metafile is an integral part of the current NCAR system, I would assume that the GKS implementation will include the metafile capability. Use of a metafile of course makes the system batch-oriented. However, a "0a" implementation of GKS should allow direct on-line output of plots even though no interaction is supported.

7. THE DI-3000 GRAPHICS PACKAGE

We have obtained the Precision Visuals, Inc. (PVI) DI-3000 package [11] for use on the display system on the VLA "pipeline." One reason for choosing DI-3000 was that it runs on the PDP-11, whereas many other available packages will not run on the PDP-11. However, even if that were not a consideration, we probably would have still chosen DI-3000 since it is one of the best packages available. Their documentation is excellent and good support is available.

It has been suggested that PVI was founded by people who left NCAR and that DI-3000 is thus derived from the NCAR Graphics Package. I doubt that this is true. The three founders of PVI previously worked on graphics at the University of Colorado Computing Center in Boulder where they produced a package called DIGRAF [12], which was an implementation of the 1977 version of the Core System. Since they have participated in the standardization efforts for many years, I am sure they were familiar with the NCAR package as well as many other packages. In any event, as mentioned above, the NCAR System Plot Package is quite different from DI-3000.

DI-3000 is a rather complete implementation of the 1979 version of the Core System. It represents more than ten man-years of effort and it includes 3D support, segmentation, on-line graphics output, metafile picture storage, and input functions. More than 50 different graphics devices are supported. DI-3000 runs on a variety of computers, including a 68000-based microcomputer.

PVI also offers a contouring/mesh surface package as a separate product. This package was written by a consultant who previously worked at NCAR, so it is probably similar to the NCAR contouring package. PVI does not currently offer any other applications-oriented packages, but there is a library of user-contributed software which can be obtained by submitting something to the library or by paying a $250 fee. They also offer some business graphics products that are probably not of much interest to us.

A recently announced PVI product is the NCAR Connector [13]. This was written by the same consultant who wrote the PVI contour package. The NCAR Connector is similar to NCAR's planned interface package except that it interfaces to DI-3000 rather than to NCAR's GKS. Also, it is available now. This package will allow the NCAR Graphics Utilities to output to any device that is supported by DI-3000. Also, it will allow on-line graphics output and use of the DI-3000 input facilities to create interactive programs that use the NCAR Graphics Utilities.

A similar interface package is available in the PVI User Contributed Library which allows existing Calcomp programs to output to any device that is supported by DI-3000.

PVI definitely plans to support GKS. They have not yet decided whether this will be an upgrade to DI-3000 or a separate product. In any event, the current DI-3000 is about 85% compatible with GKS, so use of DI-3000 should provide a good migration path to GKS.

8. COMPARISON OF NCAR PACKAGE AND DI-3000

SUPPORTED DEVICES. Before we decide to adopt a standard graphics package, we should make sure we know what device drivers are available. Also, where will drivers for future devices come from? Will they be available from the same source as the graphics package, or can we really expect the hardware vendors to provide them, or will we have to write them ourselves? At this point I don't know for sure, but I strongly suspect that the desired drivers are more likely to be available if we choose DI-3000 as our graphics package.

VIRTUAL DEVICE INTERFACE. The virtual device interface used by DI-3000 contains a relatively large set of functions. This helps allow making use of the full capabilities of newer sophisticated terminals. It also allows effective use in a networking/distributed processing environment. I do not know any details on the virtual device interface that is being planned for the NCAR GKS package.

3D SUPPORT. Neither the current NCAR System Plot Package nor the planned NCAR GKS implementation provide 3D support. If we developed any 3D plotting programs, the application code would have to handle the 3D functionality. Use of DI-3000 would be clearly more desirable since most of the 3D would be handled by the graphics package.

INTERACTIVE SUPPORT. If we adopt DI-3000, we could begin developing interactive applications immediately. If we adopt the NCAR package, it may be that some interactive applications could be developed on the VAX now, but interactive applications using GKS may not be feasible until some indefinite time in the future.

AVAILABILITY OF APPLICATIONS-LEVEL ROUTINES. The NCAR Graphics Utilities package certainly offers greater capability than anything available from PVI. Some of these routines may be very useful to us. However, if we use DI-3000 and the PVI NCAR Connector, we could use these routines with a wider variety of graphics devices, including interactive applications. Also, use of the PVI Calcomp interface would allow us to integrate existing Calcomp applications if that is deemed to be desirable.

MIGRATION TO GKS. If we choose to adopt the NCAR package, it looks like it may be at least a year before new applications could begin being based on GKS. Interactive applications could probably not begin until much later. If we choose DI-3000, new applications based on the Core System could be developed immediately. Converting these to GKS at some future time would be easier than converting applications that were based on the NCAR System Plot Package. DI-3000 would thus provide a better migration path to GKS.

GENERAL SUPPORT. The general support for DI-3000 would certainly be much better than the support for the NCAR package. PVI support includes availability of versions tailored to specific computers and operating systems, training courses, and continuing system development with automatic updates. Also, a telephone "Hotline" is available to get quick answers for problems that are encountered.

COST. The NCAR package does of course have a smaller initial cost. However, a careful analysis of all the relevant factors might show that the overall long range cost advantage of the NCAR package is not as great it may seem. The cost advantage might even go to DI-3000 in the long run. It probably would be easy to find examples where NRAO chose the lowest initial cost option but this turned out to be more espensive in the long run.

9. SOME PROBABLY IRRELEVANT COMMENTS

When the AIPS project began in early 1979, I suggested that serious consideration be given to use of a standard graphics package such as the DIGRAF package [12] or one of the public domain packages. If DIGRAF had been chosen at that point, AIPS would have had an easy migration path to DI-3000 and thus to GKS.

Although it may be feasible to run the NCAR System Plot Package on the PDP-11/44 display system on the VLA "pipeline," address space limitations would probably make it difficult to run all of the Graphics Utilities. This is probably irrelevant to the present discussions since at some point we will probably replace the PDP-11/44 by a VAX and then run AIPS on it.

10. SUMMARY AND CONCLUSIONS

I certainly think that there are advantages to using a standard device-independent graphics package. I have been saying so for five years. Eventually, everybody will probably standardize on GKS. However, implementations of the Core System will certainly be around for awhile. Adoption of DI-3000 now would allow us to immediately begin development of interactive 2D and 3D applications using a variety of devices. It would also provide a good migration path to eventual conversion to GKS. These things need to be considered in more detail before we jump to the conclusion that adoption of the NCAR package is the only way to go. If cost and/or use by other institutions proves to be an unsurmountable problem with DI-3000, perhaps we should consider one of the currently available public domain standard packages such as the George Washington University implementation of the Core System [14]. This system might also come closer to supporting the imaging functions that we need [16].

11. REFERENCES

Copies of the following are available at the VLA site:

1. Don Wells, Memorandum to Computer Planning Group, Subject: Recent Graphics Standardization Discussions, November 1, 1983.

2. P. R. Bono, J. L. Encarnacao, F. R. A. Hopgood, P. J. W. ten Hagen, "GKS - The First Graphics Standard," IEEE COMPUTER GRAPHICS AND APPLICATIONS, July 1982, p. 9.

3. "Status Report of the Graphics Standards Planning Committee of the ACM/SIGGRAPH," COMPUTER GRAPHICS, Vol. 11, No. 3, Fall 1977.

4. "Status Report of the Graphics Standards Planning Committee of the ACM/SIGGRAPH," COMPUTER GRAPHICS, Vol. 13, No. 3, Fall 1979.

5. **GKS PACKAGE.** This can be obtained by sending a self-addressed mailing label and a check for $35.00 to: X3 Secretariat/CBEMA, 311 First St. NW, Suite 500, Washington, D.C. 20001. It includes the following:
1) dpANS Graphic Kernal System (GKS) - X3H3/83-25R1
2) dpANS GKS Part 2, Section 1, GKS Binding to ANSI Fortran
3) dpANS GKS Part 2, Appendix A, Examples of GKS use in ANSI FORTRAN
4) Schedule for X3H3.5's Work on GKS (with PMIG as level m) (X3H3/83-53)
5) PMIG (Programmers Minimal Interface to Graphics) Issues Document (X3H3/83-11)

6. E. L. Sonderegger, "Comparison of Proposed 3D Graphics Standards," **COMPUTER GRAPHICS**, Vol. 17, No. 4 (October 1983), p. 250.

7. D. H. Straayer, "More Comments on GKS...," Letter in **COMPUTER GRAPHICS**, Vol. 17, No. 2 (May 1983), p. 158.

8. "SIGGRAPH Begins Steps to Adopt Core System," SIGGRAFFITI, re-prin in **IEEE COMPUTER GRAPHICS AND APPLICATIONS**, August 1983, p. 60.

9. **VDM PACKAGE.** This can be obtained by sending a self-addressed mailing label and a check for $20.00 to: X3 Secretariat/CBEMA, 311 First St. NW, Suite 500, Washington, D.C. 20001. It includes the following:
1) Review of the ANSI Metafile and Virtual Device Interface Standardization Activities (X3H3/83-73)
2) dpANS Virtual Device Metafile (X3H3/83-15 R1)

10. **NCAR GRAPHICS DOCUMENTATION.** This can be obtained by sending check for $30.00 (payable to The Scientific Computing Division of NCAR) to: Sue Long, Software Distribution, NCAR/SCD, P.O. Box 3000, Boulder, CO 80307. It includes the following:
1) An Introduction to the SCD Graphics System
2) The SCD Graphics Utilities
3) The System Plot Package
4) Selected User Reference Papers
5) The Graphics System Implementor's Guide

11. **DI-3000 USER'S GUIDE.** This can be obtained for $28.00 (plus shipping) from Precision Visuals, Inc., 6260 Lookout Road, Boulder, Colorado 80301.

12. J. R. Warner, M. A. Polisher, and R. N. Kopolow, "DIGRAF - A FORTRAN Implementation of the Proposed GSPC Standard," **COMPUTER GRAPHICS**, Vol. 12 (Proc. 1978 SIGGRAPH Conference), p. 301.

13. "Addition to Graphics Product Line," **HARDCOPY**, November 1983, p. 162.

14. J. D. Foley and P. A. Wenner, "The George Washington University Core System Implementation," **COMPUTER GRAPHICS**, Vol. 15, No. 3

(August 1981), p. 123.

15. J. Warner, "Device Independence and Intelligence in Graphics Software," COMPUTER TECHNOLOGY REVIEW, Spring-Summer 1982.

16. J. Acquah, J. Foley, J. Sibert, and P. Wenner, "A Conceptual Model of Raster Graphics System," COMPUTER GRAPHICS, Vol. 16, N (July 1982), p. 321.