THE BOLOGNA GRAPHICS LIBRARY

PRESENTATION OF THE SYSTEM

A.Ficarra and M.Nanni
Istituto di Radioastronomia - CNR

L.Federici
Dipartimento di Astronomia - Bologna University

BGL 1/1984

1. INTRODUCTION

The Bologna Graphics Library (BGL) is a software environment consisting mainly in a library of FORTRAN-callable basic graphics subroutines running under the VAX/VMS operating system.

The main purpose of BGL is to provide the user with the possibility of writing application programs designed to perform graphics operations without taking care of the characteristics of the physical device where the operations will be executed.

2. THE HISTORICAL BACKGROUND

For a long time, the researchers of the "Istituto di Radioastronomia" of CNR and of the "Dipartimento di Astronomia" of the Bologna University have made a widespread use of computer graphics in the processing of astronomical data, by both performing interactive processes on video terminals, and by generating maps on plotter-like devices.

In general, the only available graphics software was supplied by the hardware vendors, so that there were about as many graphics software packages as different types of devices installed in our system.

Some other devices were not equipped with any high level callable software and, for this reason, we had to learn how to write graphics software by ourselves. Moreover, this kind of software was always addressed to specific devices and, even if it succeeded in fulfiling the needs of the moment, it could not solve our problems in general.

Therefore, the need arose of writing a software interface that

allowed, in applications programs, the use of the same routine names and calling sequences, whichever device was selected.

As the first graphics experience we had was with the "Calcomp" plotters, we found it convenient to transport the basic Calcomp software onto the other devices, so that the source code of application programs had not to be modified or re-compiled even if a not Calcomp device was selected.

However it was still necessary to re-link every program to the driver library of the corresponding device, creating in this way as many executable images as the number of interested devices. Moreover, as the hardware capabilities, the variety of new graphics features and the user needs were more and more increasing, the old Calcomp software seemed to be no longer suitable to fulfil the new requirements.

Therefore, urged by our specific needs, we began thinking about a completly "device-independent" structure, able to provide the users with a large set of graphics functions and with the possibility of selecting any device before, or even during, the execution of a program, without need of recreating the executable image of the program.

The Bologna Graphics Library was intended to fulfil these requirements.

Obviously, we have always been conscious that all these needs were not exclusively ours and that many people in the world were interested in the same kind of problems.

In Italy, a Graphics and Pictorics Working Group was created in the ambit of the ASTRONET project, whose purpose is to promote a co-ordinated development of the national structures for the astronomical information processing. This Group organized some meetings, where stimulating discussions led to the definition of general criteria about the graphics programming methodology.

On the international ground, the most important activity in this field started in 1975 and was coordinated by the International Organization for Standardization (ISO). It led to the production of a huge graphics system, named Graphical Kernel System (GKS), which was designated as an International Standard. More than 50 man-years were invested in this enterprise. To be really a standard, GKS was designed to be not only device-independent, but also machine-independent and language-independent; moreover, it was sized to fulfil a very wide set of requirements, including those of industrial and CAD applications.

It is beyond doubt that such a system will represent the best solution for any kind of graphics problems.

In the meanwhile, we think that a much smaller system, like the Bologna Graphics Library, designed for more limited and different purposes, can still be useful, if it does not propose itself as an alternative standard, but it is specialized to work in a well defined operating system environment.

In fact we decided to restrict the BGL portability within the ambit of VAX/VMS systems. We provided the users not only with a set of FORTRAN-callable routines, but also with a software environment, strictly bound to the operating system, and allowing a large set of performances, such as, for istance, the device "spooling" management or the supporting of a suitable command language.

From another point of view, we preserved the system portability; in fact we designed the system to have a "site-independent" structure in order to be easily and quickly installed and tailored to the specific configuration of any site equipped with the same operating system.

On the other hand, the lack of "machine-independence" is not so dramatic as it seems: VAX 32-bit computers are well-known and very diffuse systems, especially in the scientific research; they are

believed to be among the most suitable systems to perform interactive processes and, in particular, graphics operations; their "Virtual Memory" feature allows to access and keep in memory large amounts of data, as it is often needed in graphics programming. Several international astronomical Institutions (such as STARLINK, ESO, VLA etc..) have adopted the VAX system as a standard of hardware and software. In Italy, people involved in the ASTRONET project suggested the same choice and, at present, all the main italian astronomical Institutes are equipped with VAX computers. So, we think that the costraint we imposed on BGL of running under the VAX/VMS operating system, is a very little limit to its portability, whereas it allows BGL to exploit very deeply the operating system capabilities.

3. SUMMARY OF BGL PECULIAR FEATURES

The remainder of this presentation paper lists the main BGL features, whose use will be explained in detail by the BGL user manuals.

- BASIC GRAPHICS ROUTINES - The main BGL system component is a library of user callable subroutines which perform basic graphics functions, such as:

drawing vectors or arcs; filling or erasing selected areas; clipping pictures inside defined windows; managing colors; saving into (or recalling from) user files, coded sequences of graphics operations; writing texts or numbers; defining set of user coordinates; executing interactive operations by means of cursors etc...

- DEVICE-INDEPENDENCE - The names and the calling sequences of the user

callable routines are independent of the device where pictures are to be displayed. Source codes of application programs remain unchanged whichever device is selected.

- "EXTENDED" DEVICE-INDEPENDENCE - The main library and the device driver libraries are linked together in a single "shareable image". An application program needs to be linked to this image only once, so creating an executable image which can run unchanged whichever device is selected.

- "DYNAMIC" DEVICE MANAGEMENT - By default, pictures are automatically displayed on the terminal screen of the user current process (provided the terminal is a graphical device). If the user wants to select another device, he (she) must explicitly give the device name, either before starting an application program, or even during the execution itself of the program: in fact, by calling a specific routine, a program can select the graphic device dynamically, according to choices made at the execution level.

- "USER FRIENDLY" AND FLEXIBLE STRUCTURE - The system was designed to make the user work as easy as possible. The routine calling sequences are mostly very simple and few arguments are usually necessary. However the system is also flexible: many routines can perform several different kinds of operation depending on parameter values. This is not a contradiction, since the system makes a wide use of defaults and optional arguments; they allow the user to avoid the specification of parameters that usually remain unchanged, unless in some peculiar cases the application problem itself needs to modify them.

We took a great care in letting the system do everything the user

does not need to do!

For example, BGL is the first graphics library where "ritual" calls to the initializing and closing routines are not required. A graphics task is automatically initialized when any routine is first invoked; and it is automatically closed when the program exits. This is possible because the system applies a peculiar VAX/VMS feature ("Exit Handler") which ensures that final operations (such as completing the last plot, or restoring the normal terminal working conditions etc..) are always executed, even if the user interrupts the program execution or an error condition causes the program to abort.

- APPLICATION AUTOMATIC UPDATING - We said above that the user callable routines and all the device software interfaces are linked together in a "shareable" image. Application programs that invoke user callable routines, address special entry-points of the shareable image, called "Transfer Vectors", which address, in turn, the actual routine locations. Each T.V. is located in a predefined and unchangeable memory position so that, even if something is modified in the shareable image, the entry-point locations of the shareable image are never modified. This means that the users have not to re-link their application programs in the event that BGL is modified, since the changes become automatically effective even for all the previously linked programs.

This feature produces very great advantages, because makes the application programs automatically updated to the current level of the BGL development, so enabling the BGL system to be more easily and more often improved and extended. In fact, we can better work to improve the system more and more, if we are free from the need of saying all the users, everytime, to re-link all their application programs.

In our opininion, this is the most important and peculiar BGL

feature.

- SOFTWARE ENVIRONMENT - The BGL system is not merely a set of graphics subroutines. Rather, it is a software environment consisting of several facilities:

First of all, a command language is available. Its purpose is to provide the user with flexible and easy means of communication with the system, in order to give or obtain information about the system itself, or to execute some operations related to the use of the BGL routines. Basically, two groups of commands are executable: the first one is reserved to the site system managers and allows to install BGL for the first time in a site, or to inform the system about changes in device configuration or in some device characteristics, or to update the system with new supplied software etc...; the second one is available to all the users and allows to obtain information about the site configuration and device characteristics, or to perform some tasks that either are related to the use of the graphics system (such as linking an application program to BGL or selecting a device different from the process terminal etc...), or execute utilities calling BGL routines (such as making a hard-copy of the screen image). By means of the command language, the user can also have access to the on-line BGL documentation. Finally, a suitable and complete on-line "help" feature make the user learn very quickly the meaning and purposes of the BGL commands and how to use them.

Moreover, a simple "data-base" contains all needed information about the site configuration and the device characteristics. This data-base is created when the BGL is first installed in a site, and it can be modified by the system manager, using the BGL command language, in the event that some changes have been made to the site configuration (for example, a new device has been installed). Everywhen an application program starts,

some BGL routines read the data-base, first of all in order to identify what device is selected, and, secondarily, to retrieve all information about the physical device parameters.

Finally, a system process manages the graphic device "spooling". This process is activated by the "startup" procedure and it is always running. Everywhen a not interactive device is selected, pictures are not displayed immediately on the device; they are written into a disk file, in a coded form. When the graphics task is finished, a BGL routine notifies the system process (by means of a "mailbox") that a new file is ready and the process takes upon itself reading the file and generating the graphic output on the device, as soon as the device is available. Then, if the operation is completed successfully, the system process deletes the file.

- SITE-INDEPENDENCE - The BGL system nucleus consists of several files containing the Fortran and Assembler coded sources of BGL programs and routines, the command language procedures, the BGL documentation and some general service modules. All references to the device and file names are present as "logical" names and no kind of information about the site configuration is directly contained in this part of the system. Thus, this part of BGL is site-independent and can be transported to any site, equipped with VAX/VMS system. All the remainder of the BGL system, that is the site-configuration data-base, the object files, and the executable and shareable images, is created at the installation time and constitutes the site-dependent part of the system. On the other hand, the use of the BGL command language makes the system installation very easy and quick.

- CALCOMP INTERFACE - We included in the BGL system a library of subroutines having the same names, purposes, and calling sequences as the

main subroutines of the basic Calcomp (and VERSAPLOT) library. These subroutines call the BGL device-independent library, working, in this way, as a software interface between the application program and the BGL system. We created this interface to allow old programs calling basic Calcomp routines to be still executable in the BGL system. Moreover, by means of this interface, also the high-level Calcomp graphics packages, that call the basic routines, are automatically usable in the BGL environment.

- ASSOCIATED LIBRARIES - The BGL system contains two more libraries of subroutines, not properly intended for graphics programming, but included in the system, because graphics routines recall them. Also application programs have access permission to these libraries; thus, BGL system can be used even for some not graphics operations.

The first library contains routines allowing to make specific input-output operations on a terminal screen, or to set some terminal line characteristics. Most of these routines are effective only on terminals of the DEC VT100 family, whereas some routines work on any kind of terminal (for example, a routine of this second group disables the terminal from receiving messages).

The second library contains miscellaneous routines; to be frequently used by the graphics libraries is their only common property. However it is worth pointing out a group of routines that can be used in several fields of application: they allow to create and manage disk files, by mapping them into the virtual memory, so making input-output operations much faster than they are in the traditional way. The BGL system uses this VMS feature to manage the configuration data-base.