technical contributions

IS PASCAL SUITABLE FOR GKS?

Sergio Antoy and Giuliana Dettori
Istituto per la Matematica Applicata, C.N.R.
Via L.E. Alberti 4, I-16132 Genova, Italy

Abstract: The compatibility between the Pascal programming language and the graphics standard GKS is considered. Both the access of GKS from a Pascal program and the implementation of GKS in Pascal language are analyzed. Various problems arise which do not seem to have simple solution.

Introduction

Recently version 7.2 [11] of the draft proposal for the ISO standard Graphical Kernel System (GKS) has been released. GKS defines a set of functions for computer graphics programming, i.e., provides a functional interface between an application program and a configuration of graphical input and output devices. The functions are organized in nine upward compatible levels with increasing capabilities. GKS defines only a language independent nucleus of a graphics system. For integration into a programming language, it has to be inserted in a language dependent layer obeying the particular conventions of that language. The description of how GKS functions are accessed by programs written in a specific language is referred to as language binding. Version 7.2 of the draft can be regarded as nearly definitive and the actual efforts toward a complete standardization concern the binding to several programming languages.

The specification of GKS is library oriented and, in most cases, the binding to an algebraic procedural language seems to be quite straightforward. However some problems arise when Pascal [2] is considered. In particular the access to every procedure implementing a GKS function and the definition of the type of parameters are crucial points of the binding. Despite some difficulties, a good number of reasons to consider and to solve the problem of GKS binding to Pascal can be found in [3,4].

GKS Used by Pascal

GKS has been designed as the specification of a graphics library, i.e., a set of procedures using a shared database. Annex C of the draft [1] specifies five rules that have to be observed to define a binding. The purpose of these rules is to make the use of GKS functions as language independent as possible. Both the number, purpose and parameters of the procedures and the content and structure of the database are completely specified by the draft proposal.

Using a library from a Pascal program is not a trivial problem since the language does not allow [2,5] a program to access external procedures. Although using external procedures might not be absolutely necessary (provided that GKS can be completely written in Pascal) a different solution seems impractical in most settings. Most

SIGPLAN Notices, V18 #5, May 1983
compilers grant the access to external procedures, moreover several mechanisms for separate compilation have been proposed [6,11], which can be used should the procedures themselves be written in Pascal. Thus using a GKS library from a Pascal program should not cause a problem. However, it relies on extensions of the language provided by some compilers, thus the language binding has to be defined as though the GKS procedures were embedded as source statements in the application program, regardless of whether they are really embedded, external or separately compiled.

The main issue in this respect is that the interface between the library and the application program can widely differ from one system to another. The differences concern the syntax of the declaration of the library procedures together with some kind of type checking of the parameters of the procedures etc. This decreases the portability of application programs thus partially frustrating an important aim of GKS.

One of the most significant aspects of Pascal, as defined by the Revised Report [2], is its type checking which has been regarded as being overly strict in dealing with arrays as parameters to procedures. For this reason the ISO proposed standard [5] considers two levels of compliance — level 0 and level 1. These differ only as level 1 allows conformant array parameters, i.e. the possibility for procedures and functions to handle arrays as parameters where the number of elements in the arrays varies from call to call.

Several fundamental GKS functions require arrays as parameters (e.g. Polyline, Fillarea etc.), thus the presence of the two levels creates a non trivial binding problem. Without conformant array parameters, application programs can become unnecessarily complicated and inefficient. Passing fixed size arrays as parameters presents serious drawbacks. Should the size of the array be too small, the activation of some procedure (e.g. Fillarea) can not be easily replaced by several consecutive activations of the same procedure (e.g. see [11] fig. 11, page 84). On the other hand a size which is big enough for every application can hardly be defined and, once a reasonable size has been fixed, it could result as too big for small systems. Thus the use of conformant array parameters seems almost essential in the binding definition. Unfortunately on some systems a level-1-Pascal compiler might not be available. It must be pointed out that GKS binding to level-0-Pascal would not result as upward compatible with binding to level 1, since the declaration of any procedure with an array parameter would differ in the two bindings. The risk that two partially inconsistent bindings will de facto coexist is another factor which, limiting the portability of application programs, should be carefully considered.

**GKS Implemented in Pascal**

Although Pascal is not intended for operating software [12,13], it offers several advantages over other languages which are available for the implementation of GKS. It is a simple, expressive, high level language offering a variety of types and structures and it allows a good design and a relatively easy maintenance of programs. However for the implementation, besides separate compilation, some features are required that are not available in the language. These regard the realization of the shared database which should be invisible to the user but global to the library and partially initialized before the application program starts.
The database is accessed by GKS functions to store and retrieve information both on the hardware configuration and on the current status of the graphics system. The scope of the variables making up the database should be limited to the whole set of GKS functions. GKS accepts the fact that a set of identifiers might be used by an implementation for internal communication. Hence these identifiers should never be used by an application program. For example in FORTRAN the set of identifiers could be implemented by means of a labelled Common declaration, thus limiting to few the number of reserved identifiers. In Pascal this is not possible and, as the number of identifiers in the database is rather conspicuous (e.g. over 30 enumeration types and over 250 variables), it is burdensome for the user. It is then convenient that the variables be invisible to the application program, that should access them only by means of GKS functions. Their behaviour must, moreover, be equivalent to global variables, i.e. their life should not be restricted to the activation of some procedure. Variables of this kind, known as static in other languages (e.g. C [14]), are not provided by Pascal. Furthermore some of these variables (e.g. Operating State) should have a precise value before the execution of the first instruction of an application program.

The above problems are sometimes solved by extensons to the language provided by some compilers which allow separate compilation [6-11]. In order to implement GKS in Pascal it is then necessary to rely on such extensions which are heavily dependent on the employed compiler causing the portability and hence the interest in the implementation to decrease.

Conclusions

In the last few years Pascal has became widely diffused due to its good characteristics. This paper explains why it is not, however, the ideal language to use with GKS. For a reasonably good utilization of GKS by a Pascal program a compiler which allows external procedures to be accessed and complies with level 1 is necessary. The implementation of GKS in Pascal presents more serious problems. A fair implementation requires a compiler with a sophisticated mechanism for the compilation of separate modules, i.e. the scope, visibility and initial value of certain variables require some kind of control that exceeds the capabilities of the Pascal language.

References


