A CANADIAN VIEW OF GRAPHICS STANDARDS

M. Wein

Electrical Engineering Division
National Research Council of Canada

ABSTRACT

Rapid reduction of costs of graphics hardware is emphasizing the high cost of producing good graphics software. One way of reducing these costs is through the development and implementation of standards for graphics interfaces.

Three aspects of graphics support software are being standardized:

(a) the view of graphics functions from a programming language.

(b) a virtual device interface which lies between the graphic support package and one of several device translators.

(c) a graphics metafile as a mechanism for storage and for transporting graphical data in a device-independent form between installations and systems.

Proposals for international standards are emerging from the USA and Germany with contributions from several countries, including a significant one from Canada.

The Canadian Working Group in Graphics within the CSA Committee on Programming Languages is participating in the development of suitable standards that would benefit the Canadian computing community.

The paper describes the current status in development and summarizes the Canadian interest in the standards activity.

RÉSUMÉ

La réduction rapide des coûts du matériel pour graphiques met en évidence les coûts élevés de production de logiciel pour graphiques de bonne qualité. Un des moyens de réduire ces coûts est le développement et l'application des normes pour interfaces de graphiques.

Trois aspects du logiciel pour graphiques font l'objet d'une normalisation:

(a) les fonctions graphiques à partir d'un langage de programmation;

(b) un dispositif d'interface pratique qui se situe entre le support graphique et un des nombreux dispositifs traducteurs;

(c) un métafichier de graphiques comme mécanisme de stockage et de transport de données graphiques sous une forme indépendante d'un dispositif entre les installations et les systèmes.

Des projets de normes internationales viennent des États-Unis et de l'Allemagne avec des contributions provenant de plusieurs pays dont une importante venant du Canada.

Le groupe de travail canadien chargé des graphiques et qui relève du comité des langages de programmation de l'ACNOR, participe à l'élaboration de normes convenables qui seraient utiles aux utilisateurs canadiens.

Le document décrit l'état actuel des connaissances et résume l'activité canadienne en ce qui a trait aux normes.

Introduction

Even though graphics is growing at a fast rate, more and more systems are becoming available and hardware costs are decreasing rapidly, the principal remaining impediment to the use of graphics is the increasingly high cost of software.

While computer software costs are high generally, the problem is more acute in graphics because of the greater portability problems. Graphics programs should not only be portable from system to system, but also be able to make use of the various displays and input devices.

Graphics standards have been developed to allow this software portability amongst devices and systems as well as a consistency of access from the various languages. After detailing some of the history of graphics standards, this paper describes the general nature of the current proposals. These proposals have become quite complex for reasons described. An alternative and simpler approach towards a lower level standard has recently emerged: The Programmer's Minimal Interface (PMI). A brief summary of the PMI is given. Also work in a related standards area - the virtual device interface/metafile is summarized.

History

Several software support packages had emerged prior to the more formal efforts in standards. Early days of graphics were dominated by batch plotting, with the defacto low level standard being the "Calcomp format" for interchange of graphical data. The emergence of the direct view storage tube (DVST) has established the "Tektronix compatible" as a low level standard. In this "Standard", graphical information is encoded in an ASCII character string on a communication channel. Tektronix has also contributed a "standard" at the software level, in the form of the PLOT 10 software support interface.

Support packages not bundled to specific hardware were developed primarily at universities, and some of these packages remain in use. The better known packages are GCS (1) (Graphics Compatibility System) developed by R. Puck, then with U.S. Army, GINO and GINO-F (2) developed at Cambridge in Great Britain, and GPGS written at the Technical University of Delft (3). More recently, the GCS has been extended to support raster and shading primitives (4).

The early groundwork for starting a more formal process towards graphics standards has been set at the Seilliac 1 Workshop held in 1976 (5) and by the Working Group 5.2 of IFIPS.

Current Proposals

The Graphics Standards Planning Committee (GSPC) was formed at Siggraph 1976 in Philadelphia (in the middle of the outbreak of Legionnaires' disease) and following a series of East / West meetings prepared the first report in 1977 (6). Subsequent effort over the next two years by several working groups with many Canadian participants has led to the publication of the final report by the GSPC in 1979 (7). The 1979 report defined the CORE proposal which has become the baseline document. It was the phenomenal growth of SIGGRAPH itself that financed the large effort needed to produce the CORE report. Defying all biological instincts, the Committee then dissolved itself with the publication of the CORE report and the formation of the ANSI Committee X3H3.

The original work on GPGS at Delft had migrated to Norway, where GPGS was rewritten in Fortran in tune with the move away from machine languages and has subsequently led to the development of IDIGS at RUNIT in Norway and which is described in an impressive and extremely readable document (8).

There was a concurrent effort on standards in Germany. Development work there has produced the Graphics Kernel System at the University of Darmstadt under the leadership of J. Encarnacao. This work rapidly progressed to becoming a DIN draft and subsequently was introduced in ISO (International Standards Organization) as a draft proposal (9). While GKS draws on much of the earlier work, it contains significant differences which will be described later (see also Ref 10).

Complexity of Standards

The common theme in all of the standards proposals is a consensus at the placement of the boundary between the standard support and the "application program". Typically, the standard provides a consistent set of

graphical primitives (primarily line, polygon, text, marker*), a mechanism for specifying primarily graphical attributes (line style, line width, polygon interior style, color, text precision, text size, orientation) and a clear definition and control of the viewing transformations that are applied to the graphical primitives. The graphical entities are defined at the standard interface in world coordinates, the coordinate system used by the application program. In most proposals, the transformation to device coordinates is applied in two steps: first, from world coordinates to normalized device coordinates (NDC), then from NDC to device specific coordinates. Thus at the NDC point in the transformation chain, the image is defined such that it is both application-independent and device-independent.

Much of the complexity introduced into a generalized graphics support system is there to support interactive graphics using either refresh or storage tube devices. The first area of both capability and complexity is segmentation, the ability to define grouping of output primitives and attributes and then to specify changes to graphical segments. The support for segmention is an important capability in interactive graphics but it comes with its inherent complexity. The complexity is compounded in generalized systems that support both refresh and storage displays. Segmentation implies buffering of the segment data. There are several strategies for updating the visible image, changes can be applied immediately and incrementally, or they can be collected or "batched" and applied all at once. Clearly, the strategy and the implementation is necessarily different for a storage display, where the screen-erase and rewrite process is far less frequent than it would be for a buffered refresh display. The graphics support standard caters to both types of displays.

Support of interaction and hence input is another important capability offered by a standard, but at a considerable expense in the complexity. The first step towards standardization of input has been the formulation of device classes, so that a measure of device independence for input can be achieved, as has been the case with output. The general approach follows the work of Foley and Wallace (11) in proposing input device classes:

Locator - - - a mechanism for specifying an xy coordinate (or xyz in 3D)
Valuator - - a device for specifying a value of a scalar function, i.e., a virtual knob
Button - - - a selection process for specifying a choice
Keyboard - - a device for entering either individual characters or an entire string
Pick - - - - a virtual device, used in conjunction with support for segmentation, for identifying segments and primitives
Stroke - - - a virtual device for specifying a series of coordinates as a path of a locator. Stroke is present in CORE but not in the GKS proposal.

The definition of virtual devices in itself does not introduce the complexity into a graphics support package. Rather, the complexity arises due to the realtime aspects of communication between the application program and the operator. The application program usually is active under the control of some operating system. It is the o/s that responds to outside events. It is therefore necessary to define a communication model that is consistent with the current practice in operating systems in order to permit a reasonable implementation of the input channel.

The interaction model defines both a simple and a complex mode for accessing input information. In the simple mode, the devices are accessed, one at a time in the form "AWAIT INPUT". In a typical time-sharing system, the execution of the application program would be suspended until operator action of the requested device. During the wait period, no other device would be enabled.

A more powerful and hence complex style of interaction permits the activation of several input devices at the same time. For this purpose, the virtual device classes are divided into two disjoint groups: those that cause events and those that are sampled. By implication, the event device implies a discrete event caused by the operator action. The sampled devices are active continuously and the application program can obtain a sample at any instant. The event devices are: BUTTON, KEYBOARD, PICK, STROKE. The sampled devices are LOCATOR and VALUATOR.

* a symbol used to label points in a graph

With the division of device classes into event type and sample type, it is possible to define a logical association between an event type and a sampled type. For example, the most common association permits the sampling of the valuator at the precise instant of an event caused by a button push. Such association eliminates the likelihood of skew that would otherwise occur if the program tried to access the two devices sequentially.

An inherent need in an interactive system is the timely response to the operator of the input actions. Input device echoing is the generalized capability for the application program and the support package jointly to respond to the operator action. Echoing, in particular good echoing, is an essential component of an interactive system. Examples of echoing are (a) the cursor on the screen indicating the (xy) coordinate being returned by the locator, or (b) a text string appearing on the CRT screen at a specified screen area, echoing the keyboard.

The standards designer is faced with trade-offs in defining echoing mechanisms. Echoing could be relegated entirely above the standard and left to the application programs. Alternately, echoing could be defined rigidly and simply within the standard. In fact, both GKS and GSPC Core include echoing with several options in their drafts because of the time critical nature of effective user-feedback.

The basic approach taken is to define a number of echo types for each device class including mandated type 0 - no echo, and type 1 - implementation-dependent echo.

In the case of sampled devices (locator and valuator), the echoing process is optionally extensible upwards at the cost of portability, in order to provide additional capability, such as rubberband echoing, segment dragging, etc., with the valuator and a similar variety of echo types for the valuator. In particular, the facility permits coupling of valuators to segment transformations - e.g. coupling a knob to control segment size.

The advantages of structured echoing are:

1. Time critical input/output coupling is kept at a low level in the system.
2. Structure in the echo definition encourages good program design practice, on the part of the application programmer.
3. Echo functions could progressively move to the graphics device in future designs.

The disadvantages are:

1. Significant complexity facing the casual user.
2. Will not satisfy all requirements.

Differences between GKS and GKPC and Impact on Canadian Activities

The GKS concept is strongly rooted in the definition of an abstract graphical work station. The abstract work station consists of one addressable display surface and a number of input devices, with at least one for each device class (10). The interpretation of graphics primitives - line-width, line style, text precision, pen-color is a two-stage process, the second stage being work station specific. In particular, the second stage of interpretation can be bundled into a "pen" which is work station specific. It is thus possible to generate the same image to produce a simplified image on a CRT and then use the data to drive a plotter work station, to produce a high quality plot. In the GSPC CORE the appearance of primitives is associated with the primitives themselves. It can be argued that CORE is more "device independent", however, GKS tries to come to grips with the essential differences between refresh displays and hard copy plotters in a way that does not require multiple generation of the image.

The position taken by the Canadian Working Group on Graphics is influenced by both the technical evaluation of the different approaches and by the practical question which of the two (if either) will become a defacto dominant standard in North America and hence in Canada.

PMI Small Core

We have discussed the complexity of the current proposals, partly due to the support of segmentation, asynchronous input, input echoing. There has been a relatively recent proposal for an extremely small and simple standard, the Programmer's Minimal Interface

(PMI), often referred to as the Small Core that attempts to limit the level of support and to minimize the complexity. Although some discussion on this topic has gone on for some time, it is only in 1980 that an effort was mounted to produce such a draft. The draft is in early discussion stage within X3H3 and while the meetings are open to public, the draft has been circulated only within X3H3 and other National Standards Working Groups.

Although the author is not part of the group working on the draft, the concept was reported to the Canadian Committee on Programming Languages and has received enthusiastic reception from the chairmen of the various language committees as well as in the Working Group in Graphics. The enthusiasm reflects the urgency for making widely available a document like the PMI much sooner than a full multilevel standard could be published. The following description is an early view by an observer of what might emerge as the PMI and is not a public statement by any group within X3H3 nor does it have any approval of X3H3.

A preliminary view of what a programmer's minimal interface might include is as follows. The PMI is a standard supporting 2D graphics in which only output functions have been included. The standard does not include support for segmentation and, therefore, it is loosely equivalent to the lowest level of the CORE.

The design philosophy has been to minimize redundant functions and to eliminate those that can be built easily on top of the PMI.

The following primitives are tentatively included in the PMI:

- Polyline
- Polymarker
- Polygon
- Text

The following attribute setting functions are tentatively included:

- Character size - set by character height
- Character rotation but only one orientation
- Text justification
- Marker symbol attribute to permit the distinguishable markers
- Color - a single index to a color table. The color model will likely be a perceptual one, based on hue, saturation, value rather than being RGB. The actual model is as yet unspecified.
- Polygon fill - fill with uniform color. No fill pattern is included.

There is no provision for grouping the attribute setting functions into virtual pens, such that a single pen function could select a number of device dependent attributes. Instead, the functions for setting attributes are available individually and are kept device-independent.

The PMI draft includes 2D window/viewport transformations but does not include clipping. Also, there is no provision of setting of background color, as might be available in hardware on a raster display.

Since the PMI does not support segmentation, it is possible to define input primitives quite separately from PMI without introducing any complexity, since it is the PICK primitive that tends to couple input and output. Without segmentation, there is no PICK function hence no coupling. If a separate input were defined, echoing of input devices could easily be built on top of PMI.

How does the PMI fit into the overall scheme? Both GKS and CORE are levelled standards. They define several levels of capability starting from output - only hard copy graphics, through to the fully interactive buffered standard with full input capabilities. At present, the specific capabilities at various levels are being studied, and one is in a position to establish Level 0 as a well defined proper subset of possibly both GKS and CORE. PMI is an autonomous definition along the lines of Level 0. As such, it meets the needs of the vast user community using hard copy graphics.

Virtual Device Interface - Metafiles

Standardization is progressing along two complementary paths. In addition to the work described earlier, a significant effort is being applied to defining a virtual device interface between the support package and device-specific translators. Such an interface, which probably will be a data driven interface, should facilitate the integration of new devices into systems. A development closely related to the virtual device interface is the definition of a standard metafile. The relationship of the metafiles to Videotex is discussed in a paper in this session (12)

One overriding motivation in developing standards in the area of metafiles is to lay the groundwork for being able to store graphical information in device independent form and to exchange graphical information between different systems and installations. There are many user communities requiring such a facility each with differing requirements. A metafile as a storage and transport mechanism can exist at any of three levels: 1. Lowest level at the virtual device interface where the information is captured for transmission to different graphics devices on the same or another system, 2. A higher level metafile which permits reading back images for further modification through interaction or processing, 3. A file exchange mechanism of application-dependent data bases. One example of the last class is an agreed standard for exchanging geocoded polygonal data for mapping purposes (13). Another example in the area of CAD/CAM is the IGES standard and its subsequent reformulations.

The publication of the Initial Graphics Exchange Specification (IGES) in January 1980 (14) has created much interest in the CAD/CAM community by promising a timely mechanism for the exchange of product definition data in machine readable graphical form. Subsequently, the document has been incorporated into a draft by the ANSI Committee on standards for machine readable engineering drawings - Y14.26.

The IGES draft is based on two corporate standards for product definition data bases: the Boeing CIIN and the General Electric Neutral Data Base. Being primarily intended for data base exchange, it is not a "graphics" standard. The intent of IGES lies clearly outside of the intent of graphics standards established at Seilliac I (5) that application dependent modelling functions are beyond the scope. Furthermore, the effort is clearly in the mechanical engineering community.

Regardless of any other work in the area of strictly graphical metafiles, the IGES effort will have a significant impact in that sector of graphics devoted to CAD/CAM.

Conclusion

The field of graphics standards has shifted from the forum of technical societies to the formal standards developing organizations - CSA, ANSI, ISO, DIN. The proposals are now being studied from a broader point of view than formerly. Clearly no firm conclusions can be drawn at this time. However, it is expected that by late in 1982, draft standards will reach a degree of maturity that implementations will appear on the market. Actual generality of their use must result from the support of the vendor/user community which will depend on the soundness of the proposals. It certainly will not be dictated by any standards writing committee.

References

(1) R. F. Puck, Graphics Compatibility System Ref. Manual U.S. Army Waterways Exp. Stn. Vicksburg, MS Oct. 1977

(2) P. A. Woodsford, The Design and Implementation of the GINO 3D Graphics Software Package. Software-Practice and Experience V1, pp 335-365 (1971)

(3) L. C. Caruthers, et al, A Device Independent/General Purpose Graphics System for Stand Alone and Satellite Graphics Computer Graphics V11, N.2(1977) p.112

(4) G. Laib, et al, Integrating Solid Image Capability Into A General Purpose Calligraphic Graphics Package. Proc Siggraph '80 p79 Computer Graphics V14, July 1980

(5) R. Guedj (Ed) Proc. Workshop on Graphics Standards Methodology (1976) Seilliac France

(6) GSPC Status Report of the Graphics Stan dards Planning Cttee. of ACM/SIGGRAPH

(7) GSPC Status Report of the Graphics Standards Planning Cttee. of ACM/SIGGRAPH Computer Graphics Quart. V13,N3 (1979)

(8) Ketil Bø IDIGS Interactive Graphics System Vers.3 Runit, Norway (1980)

(9) Graphical Kernel System GKS, Functional Description Vers. 6.4, Working Draft ISO/TC97/SC5/WG2

(10) J. Encarnacao, et al, The Workstation Concept of GKS and The Resulting Conceptual Differences to the GSPC Core System Proc. Siggraph '80 p.226

(11) J. D. Foley and V. I. Wallace, The Art of Natural Graphic Man-Machine Conversation Proc. IEEE V.62, No. 4 (1974) p.462

(12) H. Newman, The Relationship of Telidon and Computer Graphics Standards. This vol.

(13) Standard Format for the Transfer of Geocoded Polygon Data. Spatial Data Transfer Cttee.,Dept. of Energy,Mines and Res.(1979)

(14) R. Nagel, et al, Initial Graphics Exchange Specification IGES Vers. 1. Nat'l. Bureau of Standards, USA NBSIR 80 -1978(R)