Session 4

GKS - THE FIRST GRAPHICS STANDARD

F.R.A. Hopgood

1. Introduction

Standards in computer graphics are long overdue. Whereas de facto standards in programming languages were common very early on (FORTRAN and ALGOL 60) and international standards soon followed, there has been a long period of graphics history where, at best, regional de facto standards have existed and no international standards have evolved.

1.1 Early History

It may be thought that this is due to the relative youth of the subject area but this can soon be dispelled. Dual 16 inch displays were available on the MIT Whirlwind as early as 1951, plotters were in regular use by 1953 and high speed microfilm recorders such as the SC4020 were available in 1958. It is true that the more exotic input peripherals and higher quality displays did not appear until later. Even so, lightpens existed as early as 1958, the RAND tablet in 1964 and colour displays appeared in 1962. By 1965, most of the hardware facilities used today were already in existence.

One clue to the delay can be seen if we look at the number of displays installed world-wide. A recent survey indicated that as late as 1964 there were only about 100 refresh displays in the world. These were developed by a number of different companies with quite different order codes and inevitably different graphics packages were developed that were oriented towards a particular display. To some extent, the same occurred in the early history of programming with languages such as Mercury Autocode and Jovial being developed. However, the availability of FORTRAN on a range of IBM machines which sold in large quantities was the main reason for the appearance of FORTRAN as a de facto standard.

By the late 1960s, common techniques for use with stand-alone or satellite refresh display systems were beginning to appear. It is possible that if graphics had continued developing in this way, standards would have appeared. Similarly, there was beginning to appear a certain amount of standardisation in the packages used for plotters. For example, GINO and GHOST began to gain acceptance in the UK in the early 1970s.

1.2 Time Sharing and Storage Tubes

Two factors emerged which tended to upset the conventional way of working and significantly changed the population of graphics users. Until the late 1960s, the only way to do interactive graphics was to use a dedicated host computer with an expensive refresh display. A typical system might cost as much as $400K. However, by the late 1960s time sharing systems began to appear which allowed a large user population to use a mainframe host interactively. Project MAC at MIT is a good example of the pioneering work in this area. This alone would not have made a large impact due to the high cost of displays (typically $80K at that time). However, at approximately the same time, the Tektronix storage tube appeared which was only a fraction of the cost of the refresh display (typically $4K when it first appeared). Consequently, the door was open for a large number of people to do interactive graphics. These users had little in common with the earlier users of interactive graphics and developed new techniques more applicable to the new storage tubes. The device itself was unable to do immediate updates without completely redrawing the picture and the slow line speeds to the host computer made redrawing unattractive.

1.3 Current Situation

To some extent, this continual change in the economics of computer graphics is still with us. Over the last few years, the appearance of low cost raster displays and plotters (Versatec etc) has meant that the storage tube now has a low cost rival capable of mimicking the storage tube but also being capable of selective erasure and additional functions such as area fill. Low cost colour displays and plotters are beginning to appear so that no longer is it feasible to ignore colour graphics as an expensive luxury for the few.

1.4 The Future

The question may be asked 'Is now the time for standardisation' or are more advances over the horizon that will completely change the way of working. We shall probably not get the same rapid changes in usage and hardware in the future. One change which is clearly coming is the appearance of high powered single user systems at low cost. Effectively, we have come full circle and the dedicated graphics workstation that started the graphics evolution may triumph as the standard future workstation. We will see changes in the input/output devices but they are more likely to be orthogonal to the current set. In particular, voice input/output will clearly have an effect in the future. The advent of low cost devices controlled by microprocessors gives us the opportunity to put more functionality in the hardware. The main question is what functionality do we firm up in the hardware. The appearance of standards for computer graphics clearly will help in this area.

2. SEILLAC I

Most of the current activities in standardisation have their origins in a Workshop organised by IFIP WG5.2 in Seillac, France. At an IFIP WG5.2 meeting in Malmo, Sweden in August 1974, Richard Guedj was asked to initiate an active programme directed towards establishing standards for computer graphics. A Committee was set up to consider what should be done. After several meetings, this Committee, consisting of designers, implementers and users, came to the conclusion that no known computer graphics system could be considered suitable as a basis for a standard. Furthermore, it was realised that a major problem was the lack of a clear understanding of many of the major concepts involved in graphics systems. Consequently, after a meeting at Bellinglise, it was agreed that it would be premature to initiate specific recommendations on standards but, instead, a workshop should be organised of experts in the computer graphics field to see if the underlying concepts of computer graphics could be uncovered. This Workshop, later given the name Seillac I, took place in May 1976 (1). A subsequent workshop looked specifically at the problems of interaction and was given the name Seillac II (2). The intention is to hold a third workshop when the time is right to consider the next bottleneck to our understanding in the area of man-machine interaction. It is still unclear what that topic should be especially as we are still digesting the conclusions of Seillac I and II. The importance of Seillac I did not become clear until considerably later. Originally, there had been no intention to publish the proceedings but, at the urging of IFIP, the working papers were edited nearly two years later and published. The Seillac I volume is, therefore, not a polished document but it gives the seeds of future activities.

2.1 Motivation for a Standard - Why, What and How

Some time was spent at Seillac I looking at the reasons for standardisation. It was generally agreed that a major reason was that standardisation improved communication from the user point of view. The ability to move software from one installation to another was of fundamental importance for a wide range of users. Portability meant that software costs reduced and the cost of training personnel decreased. From the point of view of the manufacturer, improved portability increases the size of the market. The standard itself gives guidance as to the right directions for hardware innovation.

The scope of a standard was less clear in 1976 than it is today. Considerable discussion was had over whether the standard should restrict itself to just considering output which was reasonably well understood and ignore interactive graphics until more work had been done in understanding the basic concepts of interaction and, in particular, the overlap with psychology and skill acquisition. The view of the majority was that standardisation of output would be easy and input should also be considered.

The requirements of a standard were also considered in some detail. It was clear that a well defined area of applicability needed to be agreed. While most people accepted that, say, schematics, engineering drawing, cartography and animation should be valid areas in which to use the standard, it was less clear that the more fringe activities of image processing and high quality typesetting should be accommodated. A standard must meet the needs of the user population (once that has been established) and should rationalise current practice as well as providing a significant advance over existing systems. Widespread acceptance is only likely if the standard is defined with a high level of expertise (a poor standard is worse than no standard) and it must not conflict with other standards in related areas (character sets, communications, programming languages).

2.2 Interface Level

The standard must be defined at such a level that it meets the needs of most users. Standardisation at the application level, for example, may be highly desirable but may not be of general use. For example, ORTEP is a standard for molecular drawing but it has little applicability outside its own field. If the standard is at too low a level then it will become device dependent and suffer for similar reasons.

Two candidates for standardisation levels are the user level and the code generator level (dividing line between device-independent and device-dependent often called DIDD). The user level corresponds to the graphics subroutine package such as GINO-F while the DIDD level corresponds to the GINO-F device driver interface.

Subsidiary standards are also required. A major one is the metafile for storing or transmitting information from one location to another. Closely associated with it are the relevant communications protocols for the transmission of graphical information. Again, metafiles can be defined at a variety of levels from the DIDD to the virtual terminal level.

2.3 Virtual Devices

By 1976, it was generally accepted that any standard would not interface directly to specific devices but would define a set of virtual input and output devices or primitives which would be simulated by actual devices (3).

It was generally agreed that line drawings and text output would be essential primitives of any standard. There was also strong support for a primitive to mark a point (later called marker). There was less agreement over the form and attributes to be associated with each primitive.

On the input side, STRING, LOCATOR, PICK and VALUATOR were accepted as virtual input devices with the following characteristics:

STRING	:	input a text string
LOCATOR	:	input a position on the display
PICK	:	input the name of an item pointed at on the display
VALUATOR	:	input a real value

There was some discussion over the need for additional virtual input devices. Strong candidates were a CHOICE device which simulated the input of a logical value using pushbuttons and a STROKE device which input a complete set of positions - particularly useful for digitisers and free-hand drawing.

Over the years, changes in hardware and usage have seen the appearance of new virtual devices and primitives together with modifications in the meanings of the original definitions. However, by and large, the virtual device model used today closely resembles the proposals put forward in 1976.

2.4 Current Position

A major discussion arose at Seillac I over the specification of line drawing output and the use of coordinate systems in existing computer graphics packages. Almost invariably, the concept of a current position existed in most packages and almost certainly it arose from the plotter or display hardware which, after some output command, left the pen or display beam at the last drawing position. The standard primitive for drawing a line was of the form:

```
DRAW(X,Y)
```

which drew a line from the current position (XC,YC) to the point (X,Y) and then redefined the current position to be the point (X,Y).

Also, facilities existed in most packages for redefining the coordinate system being used by the applications programmer. He would define a coordinate system applicable to his application and define a rectangular area in that coordinate space (WINDOW) to be mapped onto a particular area of the display device (VIEWPORT). Changes in coordinate systems were used for two quite distinct purposes. The first effectively provided a different view on the display of the world scene while the second allowed graphics information to be moved about the screen to compose a picture. Often, the same routine in the package was used for both purposes frequently causing confusion as to its real meaning. The problem can be typified by this simple example:

```
TOXY (0,0)
DRAW (1,1)
TRANS (3,3)
DRAW  (2,2)
```

The TOXY command defines the current position as being at the origin of the defined coordinate system. The first DRAW command draws a line between (0,0) and (1,1). The TRANS command redefines the coordinate system so that the new origin is at the point (3,3). The final DRAW command draws a line to the point (2,2) from the current position. But where is the current position? It was left at (1,1) by the previous DRAW command. At this stage, it is necessary to take a closer look at the meaning of current position. If current position is a point on the display screen, the new line will join up with the old. If it is a syntactic abbreviation to save writing out parameters, a line from (1,1) to (2,2) will be drawn in the new coordinate system. Does it make sense to draw a line between a point defined in one coordinate system and a point in a different coordinate system? At some conceptual level, it is clearly illegal and the change of coordinate system should effectively destroy the current value of current position.

Packages existed with all variants discussed above and some not described. The major problem was that existing packages had no clear distinction between coordinate transformations being used for altering the viewing of a picture and those used for modelling a picture out of smaller items where the coordinate transformations are moving sub-parts of an object on the screen to compose a picture.

A major resolution at Seillac I was that there should be a clear distinction in any standard between those parts that dealt with modelling and those parts dealing with viewing. Furthermore, an initial goal should be to define a core graphics system for viewing aspects of a picture already constructed in world coordinates.

3. GSPC AND GKS

Participants at Seillac I included Peter Bono, Jim Foley and Andy van Dam of the USA and Jose Encarnacao and George Nees from Germany. Jim Foley had been a founder member of a Graphics Standards Planning Committee (GSPC) which had been formed after a Workshop on Machine-Independent Graphics organised at the National Bureau of Standards in 1974. Work by GSPC had proceeded slowly until Seillac I but the enthusiasm generated at Seillac led GSPC to work towards the specification of the core graphics system (identified as a major goal at Seillac) as soon as possible. A considerable amount of work was put in by GSPC under the auspices of ACM-SIGGRAPH which culminated in two main drafts of a core graphics system GSPC77 (4) and GSPC79 (5). A good overview of the GSPC core system is given in an issue of ACM Computing Surveys (6). GSPC79 is a full 3D core system and a number of implementations of different degrees of completeness have been produced in the USA.

At the same time as the USA group were defining the GSPC core system, the German DIN group were working on the Graphical Kernel System (GKS) which was also aiming to define a core graphics system for viewing. One major difference between the two proposals was that GKS was only a 2D system and, initially, was significantly smaller than the GSPC core system.

4. ISO

Some time in 1976, the Standards Committee of the British Computer Society proposed that GINO-F should be put forward as an international standard to ISO. The relevant ISO group, ISO/TC97/SC5, had no appropriate working group to consider such a proposal and so a working party was organised by SC5 in London in February 1977. This meeting of experts examined the draft of the GSPC core proposal, had presentations from a number of experts, and came to the conclusion that no existing graphics software package could be considered suitable for recommendations as a graphics standard. The meeting recommended to SC5 that a working group (later called WG2) should be set up to review all material issued by GSPC, IFIP etc in the general area of computer graphics standardisation. It urged that an early specification of a core graphics system was desirable.

4.1 Toronto and Bologna

The inaugural meeting of ISO/SC5/WG2-Graphics was held in Toronto in August 1977. The major item of discussion was the latest version of the GSPC core project. However, other countries mentioned their standards activities and, in particular, Jose Encarnacao for the German DIN group indicated that DIN WG5.9 had been working towards a functional specification of a core graphics system to be proposed as a German standard. A major resolution of the Toronto meeting was that DIN and ANSI should work towards a common specification of a core graphics system.

The next WG2 meeting was in Bologna in September 1978. No official USA representative attended, although Ketil Bo, who was resident in the USA at that time, and Robin Williams of IBM acted as reporters of GSPC activities. The DIN group presented the current state of GKS and outlined timescales that would lead to a DIN standard by 1981. Norway also indicated that they would be proposing IDIGS, a successor to GPGS, as a Norwegian standard. The working group recommended that an Editorial Board of WG2 should be set up to compare the various proposals for standardisation, note the main areas of difference and recommend changes so that the three proposals would converge towards a single draft standard proposal.

4.2 Editorial Board - Amsterdam 1979

The Editorial Board met in Amsterdam in February 1979 (7). The expected IDIGS proposal did not appear in time and the meeting compared GKS Version 4 and GSPC77 attempting to identify major differences in concept and to point out minor changes that would bring the two proposals closer together. A fundamental difference between the two proposals was the lack of a current position in GKS and the concept of a pen in GKS which could have quite different attributes associated with it on different workstations. Thus a pen could be a green thick line on one workstation while it appeared black and dotted on another. The GSPC proposal had a more conventional set of model attributes such as colour and linestyle which had to be simulated by the implementor in the best way possible on a specific device. The Editorial Board recommended a number of changes to both GKS and GSPC which would bring the two proposals closer together.

Both DIN and GSPC discussed the Editorial Board recommendations and a joint meeting was held in Boulder, Colorado. By June 1979, it was decided that the GSPC work should be wound up and passed over to ANSI X3H3, the formal standards body in the USA equivalent to DIN and BSI.

4.3 Budapest 1979

The next ISO meeting took place in Budapest in October 1979 where DIN presented a new version, GKS 5.1, which incorporated a large number of the recommendations made by the Editorial Board. Much richer input facilities were included and the ability to have several displays active under the control of a single operator. ANSI presented GSPC79 which included a pen concept on the lines of GKS and text output had been changed extensively to allow quality text production which was becoming more and more important as the influence of business graphics expanded. A presentation was also made of IDIGS.

The DIN group were keen that GKS should be submitted to ISO as a standard proposal. On the other hand it was possible that standard proposals would appear from the USA and Norway in due course, although neither were at the same level of technical refinement as GKS at this point in time. There was some discussion as to whether it was feasible for the working group to evaluate two proposals in parallel. Eventually it was decided that only GKS would be put forward to ISO with the aim of GKS reaching the level of a draft proposal in one year. Later events showed that this was over optimistic!

4.4 Tiefenbach 1980

It was agreed that a Technical Meeting should be held in Tiefenbach, Germany in June 1980 before which member nations would do a thorough review of GKS bringing forward outstanding issues that needed to be resolved with all alternatives and arguments for and against fully documented. The major input to the Tiefenbach meeting was the ANSI group who put forward over 200 issues while BSI and other national bodies also contributed significant numbers of issues.

Issues tended to fall into a number of classes ranging from clarification of the current version of GKS to proposals aimed either at reducing the size of GKS or increasing its functionality. The major set of issues resolved at Tiefenbach were in the area of clarification. Thus, although on paper it appeared that the number of issues resolved was high, the main substantive issues remained unresolved. In the area of viewing, there was still discussion as to whether a shielding facility should be built into GKS and whether clipping should be defined by the bounds of the window or by a separate rectangle. The quality of the text output increased significantly after Tiefenbach with the ability to define character orientation being added. There were still unresolved issues such as should proportionally spaced fonts be provided and also text alignment. There was a great deal of discussion as to whether locator input should be returned in world or device coordinates. GKS 5.2 was used as the basis for the discussion at Tiefenbach although DIN had produced a new version GKS 6.0 which incorporated many of the ANSI suggestions. The result of Tiefenbach was GKS 6.2 and it was agreed that this would be the basis for the next round of technical discussion at Melbourne, Florida with an editorial round proceeding in parallel to improve the quality of language used and to adhere to the rigid formats defined by ISO for a standard document.

4.5 Melbourne 1981

The major contributor to the Melbourne meeting in January 1981 was BSI. There was still considerable disagreement as to the form that input should take in GKS and this area continued to have the most unresolved issues after Melbourne. The BSI proposed an extension to the window/viewport definition to allow multiple window/viewports to be active at one time. This indirectly made it much easier to return locator positions in sensible world coordinates. The text primitives became more exotic as a result of Melbourne allowing the direction of text to be specified as well as orientation. There was considerable discussion over whether attributes should be defined by the GKS pen concept or by modal attributes. A much cleaner pen facility was established in GKS. Considerable discussion took place around the raster primitives and fill area. The general direction was to provide more facilities for patterning area fill and then query whether pixel array was still a valid primitive especially as some operations such as rotation did not easily get applied to pixel arrays.

4.6 Abingdon 1981

The decision was made at Tiefenbach that two more rounds of technical comment would take place before the ISO TC97/SC5 meeting in London in October 1981. The final technical meeting was organised at Cosener's House, Abingdon, just prior to the SC5 meeting where the aim was to resolve all remaining issues, rewrite the document and present it to SC5 as the first draft standard proposal the following week. The main areas of discussion were input, text, segmentation and levels. A simpler input model was agreed upon which had the capability of being extended to provide more exotic facilities on top. The text facilities were extended even further and it was now possible both to use the capability of hardware character generators and to produce quality text by software. Considerable discussion occurred around the INSERT SEGMENT facility which provided a degree of modelling within GKS. A more precise description requiring changes to the implementation technique was defined which also improved the characteristics of general segment transformation.

All remaining issues were resolved at Abingdon with only two facilities remaining outside GKS although carrying significant support. These were text alignment and the possibility of adding a STROKE input primitive. It is likely that these functions will be raised again at later technical discussions after the draft standard has been accepted.

On 9 October 1981, GKS 6.8 (to be GKS 7.0 after further editorial work) was accepted by ISO TC97/SC5 as a draft proposal (DP). Because full agreement had been reached within WG2 about the technical contents of GKS, it was recommended that SC5 should circulate the GKS-DP for letter ballot for approval as a Draft International Standard. Six years had elapsed since the meeting in Bellinglise and over five years from Seillac I where Richard Guedj opened the Workshop by quoting:

> I have long aspired to reach for the clouds ...
> Again I come from afar
> To climb Ching Kang Shan, our old haunt
> But scenes are transformed

Mao Tse Tung, May 1975

5. GKS

GKS had a long and intensive technical review which benefited enormously from the earlier work of GSPC and the meeting at Seillac. The final form of GKS can truly be claimed to be an international standard. The influences of many national bodies can be seen in GKS. It differs quite dramatically from the earlier versions and yet its main concepts are still very evident and, if anything, have been clarified and strengthened over the period. There is an almost irresistible urge after this length of time for compromises to be made so that the end result bears a close resemblance to a camel. That GKS has survived the many redrafts and still clearly exhibits its main methodology is to be commended.

5.1 GKS Workstation

The main objective of GKS was to allow easy portability of graphics systems between different graphics installations. Although GKS should be capable of being used in small stand-alone graphics programs, it was also essential that large CAD suites of programs could be written and moved from one installation to another, possibly with quite different hardware, without modification of the program structure.

It was likely that the characteristics of different graphics displays would differ significantly and that it would be difficult to model a wide range of facilities with close approximations on all devices. It was felt that implementation defaults for unavailable facilities would not always be applicable and there was a need for the applications programmer to have some control over the mapping of graphics primitives to a particular device.

Central to GKS is the concept of a graphics workstation with a single display area and a number of input devices. It is assumed that the workstation has a certain amount of intelligence either local to the display or in the workstation driver. The workstation is defined as belonging to one of a set of standard types (plotter, storage tube, refresh display etc) with the ability for the applications programmer to modify its overall behaviour to fit in with the application area.

An operator can have a number of GKS workstations under his control at the same time. For example, he may be outputting a large CAD drawing on a plotter while getting a quick-see view on a separate storage tube. He may be interacting at a refresh display taking occasional copies of output on a plotter.

The applications programmer has considerable flexibility in how he uses each workstation. Different workstations may be set to view different parts of the whole graphics picture. The frequency of update may be different on different devices.

Moving from one installation to another will cause the applications programmer to redefine his workstation definitions. These would normally be defined at the start of a program and should not alter the main control flow. For example, moving from a Tektronix 4010 to 4014 might cause the area of the picture in view to be expanded.

The lack of a tablet may necessitate a different method of entering locator positions. The type of echoing may depend on the line speed between display and computer.

5.2 GKS Pens

Graphics primitives such as line drawing can have attributes associated with them such as colour, thickness, broken, etc. There are basically two approaches to specifying such attributes. The first is to have a set of modal attributes which are in effect until the next setting of the attribute. This is the conventional method of specifying attributes and is used in the GSPC core system. For example:

```
COLOUR(RED)
THICKNESS(THICK)
BROKEN(SOLID)
DRAW LINE
COLOUR(GREEN)
BROKEN(DASHED)
DRAW LINE
```

This would draw a thick, red, solid line followed by a thick, green, dashed line. The particular modal attribute remains in effect until it is reset. Thus solid is an attribute applicable to both lines.

A disadvantage of this approach is the need to map this attribute specification on to a number of devices that may not have the capability to implement a particular attribute. How do you draw red lines on a storage tube? It is usually left to the implementor of the device driver to make an arbitrary decision. A second disadvantage of this approach is the specification of library routines where differentiation of particular lines is required but it would best be left to the application programmer to specify the particular attribute to use. For example, a contour routine might wish to highlight every third contour. The application programmer might wish to use colour, thickness or broken lines to highlight the effect. With modal attributes, the body of the algorithm becomes quite complex with many attribute settings depending on the user's requirements.

The solution adopted in GKS is not to have a number of modal attributes but instead to have one major attribute per primitive called the pen number. Each primitive may have one of a number of pens associated with it running from 1 up to an implementation maximum. The equivalent GKS program to the one above would look like:

```
PEN(1)
DRAW LINE
PEN(2)
DRAW LINE
```

On a particular device this would draw the first line with pen 1 and the second with pen 2. The definitions of pen 1 and 2 are workstation dependent and can be set by the application programmer. Thus, he can set pen 1 as red, thick and solid while pen 2 is green, thick and dashed. The advantage of making the pen specification workstation dependent is that the characteristics of pen 1 and 2 can be quite different on two workstations. For example, a user with a large plotting table and a storage tube to give him an overall view of the plotting could specify colour as the main attribute on the plotter while defining the different pens as different types of dashed lines on the storage tube.

5.3 Output Primitives

GKS has defined six output primitives:

(1) POLYLINE
(2) POLYMARKER
(3) TEXT
(4) FILL AREA
(5) PIXEL ARRAY
(6) GENERALISED DRAWING PRIMITIVE (GDP)

A major feature of GKS is that is has no concept of current position. Each primitive has its coordinates fully defined within the primitive itself. Furthermore, in the case of line drawing, a polyline, which generates a set of connected lines given an array of points as parameter, is the fundamental line drawing primitive. The motivation for this is that very rarely are single lines drawn. Instead, it is more common to output a set of lines to form some shape. Given that polyline rather than line is the basic primitive, attributes such as broken apply to the complete polyline rather than a single line segment.

Thus, dotted or dashed curves are easily drawn.

Polymarker is an obvious primitive once polyline has been defined. Text similarly produces a string of characters rather than a single character so that there is a degree of similarity of level between the three main primitives.

The remaining three primitives are likely to be less commonly used but show the influence of raster graphics and the need to allow expensive hardware facilities to be used even within a standard. Fill area defines a boundary which will be drawn and its interior can be filled in with either a pattern or a hatching. Other functions define the form of the shading. Pixel array is a means of specifying an array of pixels and is particularly of use in image processing. The final primitive, GDP, is an escape function to allow special primitives such as circle or curve to be defined in a well defined implementation specified way - a standard way of being non standard!

5.4 Attributes

Polyline and polymarker have a single attribute, the pen number, which can be defined as follows:

POLYLINE	:	LINETYPE,	LINEWIDTH	COLOUR
POLYMARKER	:	MARKERTYPE,	MARKERSIZE,	COLOUR

Text on the other hand has two sets of attributes, some of which are set by the text pen table:

TEXT : FONT, PRECISION, COLOUR

while the geometric attributes are set modally. The motivation for this split is that the overall form and shape of the text must fit with the graphical output on all devices and so should be device independent while the particular character forms and quality of characters drawn may differ from workstation to workstation and should,therefore, be part of the pen table.

The modal text attributes are:

1. HEIGHT: defines the required height of the character in the user's coordinate system.

2. EXPANSION FACTOR: the font has a specified height/width ratio. EXPANSION FACTOR defines how much larger than the specified width it is.

3. CHARUP VECTOR: defines the direction that the character height should take. Characters can, therefore, be drawn at any orientation.

4. PATH: defines the direction in which characters are drawn. The normal setting is LEFT while RIGHT draws them from left to right. Similarly, UP and DOWN have their obvious meanings.

5. SPACING: additional space between characters.

It is recognised that some devices may have difficulty specifying characters to that degree of sophistication. Consequently, the PRECISION attribute in the text pen table defines the closeness of the output to the specified requirements:

1. STRING: the position of the first character is all that is guaranteed to be correct. Thus, a device's hardware character generator can be used. If a different orientation or size is requested, it can be ignored.

2. CHAR: the positions of the individual character boxes must be correct. The form of the character within the box is workstation dependent. Again, hardware characters could be used but they would probably have to be output one at a time.

3. STROKE: all the text attributes have to be implemented correctly. This will almost certainly require the hardware to have a very flexible character generator or the text output to be simulated in software using polylines or fill area primitives.

The current method of defining text in GKS does make it possible for sophisticated hardware character generators to be used if available. On the other hand, the workstation can choose a much simpler representation for all but STROKE precision text.

5.5 Segments

It is possible to call output primitives so that they are displayed on all active workstations. However, in an interactive environment, it is frequently important that the complete display can be split into a number of sub-objects or segments which can be manipulated independently. You may wish to highlight a particular part of the display or remove it for some reason. In working with a refresh display, it is frequently required to move parts of a picture around. This is achieved by having a transformation matrix associated with a segment which may be altered after the segment is defined.

Segments are stored on the workstations that are active when the segment is defined. This is adequate for most purposes but occasionally you need to have a segment appear on a workstation that was not activated when the segment was created. For example, the user may be defining a picture made up of segments on a refresh display and then at some stage he may wish to copy the current display to a plotter. This is achieved in GKS by having a device independent segment storage which can keep copies of segments as they are formed and apply the same transformations as are applied to the segment on the workstation. When a copy is required, the segments can be sent from device independent segment storage to a specified workstation. Facilities are also provided for INSERTing a segment into another segment. This modelling facility is only allowed in the more complex GKS implementations.

5.6 Viewing

The standard graphical package tends to have a window/viewport transformation which allows the applications programmer to define his own coordinate system, some parts of which are mapped on to an area of the display screen. The situation is complicated in GKS by having several workstations active at the same time. Is it sensible that all workstations are forced to use the same viewport on to the world scene? An application might require one display to give an overall view of the picture being displayed while another looks at the detail of the picture.

This flexibility is achieved in GKS by having three different coordinate systems and two distinct window/viewport mappings. The applications programmer defines his output in terms of a world coordinate (WC) system which is mapped on to some part of the normalised device coordinate (NDC) plane. The set of active workstations can then take separate views of the NDC space mapping these on to workstation dependent parts of the display.

For any complex picture, it is likely that it will be made up of several distinct parts which are most appropriately defined in different coordinate systems. A conventional package would do this by allowing the user to continually redefine the window/viewport mapping from world coordinates to NDC. For example:

```
SET WINDOW(XMIN,XMAX,YMIN,YMAX)
DRAW PICTUREA
SET WINDOW(X2MIN,X2MAX,Y2MIN,Y2MAX)
DRAW PICTUREB
```

Here, PICTUREA is drawn when the first coordinate system is defined while the PICTUREB is drawn with the second coordinate system. The user effectively sees a display made up of two parts with different coordinate systems. The user's view of the system is that both coordinate systems must be known to the system as pictures are being displayed with respect to them. However, in reality, only the latter of the two coordinate systems is currently known. This often causes confusion as the user expects to be able to point to a particular position in either coordinate system and the system should deliver the position in the correct coordinate system.

To ensure that the user's view of the system is the correct one, multiple window/viewports can be defined in GKS and they are all in existence at the same time. The equivalent form of the above program in GKS would look like:

```
DEF WINDOW (1,XMIN,XMAX,YMIN,YMAX)
DEF WINDOW (2,X2MIN,X2MAX,Y2MIN,Y2MAX)
SELECT (1)
DRAW PICTUREA
SELECT (2)
DRAW PICTUREB
```

Note that the form of the program in GKS will have a tendency to define all the coordinate systems required at the start of execution and then select the particular transformation as and when required. The other program form will have transformation definitions scattered throughout the program.

5.7 Input

Input in GKS is defined in terms of a set of logical devices which may be implemented on a workstation in a number of ways. The different types of input are:

(1) LOCATOR: provides a position in world coordinates. The position indicated on the display will be within one of the window/ viewport transformations defined. This will be used to give the correct world coordinate position.

(2) VALUATOR: provides a real number.

(3) CHOICE: provides an integer defining one of a set of choices.

(4) PICK: provides a segment name and a pick identifier associated with a particular primitive.

(5) STRING: provides a character string

The implementation of the logical device on a workstation may be done in a variety of ways. For example, it may be natural to input a STRING using a keyboard, it could also be done by free hand drawing on a tablet or by hitting a set of light buttons indicating particular characters on a display. The exact form of the implementation is up to the workstation.

Input can be obtained in three distinct ways:

(1) REQUEST: this is rather like a FORTRAN READ. The system waits until the input event has taken place and then returns the appropriate value. Only one input request is valid at a time.

(2) SAMPLE: the current value of a GKS input device is examined. Most frequently used for devices which have a continuous read out of their value. For example, the current position of the pen on the digitiser can be sampled or the position of a potentiometer.

(3) EVENT: this mode is used for devices which would normally cause interrupts on the workstation. For example, a light pen hit or pressing the tip switch on a tablet would normally generate an event. Such events are stored in a queue in the order they arrive and functions are provided to take events off the queue and deal with them.

Earlier versions of GKS had a much more complex input system with non-sequential dequeueing. It was decided that such functions should be built on top of GKS rather than be part of the kernel system.

6. SUMMARY

This paper has given a summary of the main events leading up to the appearance of GKS 7.0. The description of GKS in the previous section is intended to give a flavour of the facilities and methodology of GKS. It does not cover all features and oversimplifies many of the facilities. No attempt has been made to describe the GKS level structure which allows only some of the functionality to be available in a particular implementation. Nor has any attempt been made to describe the more device dependent features which allow efficient working on plotters and storage tubes.

As the first international standard in computer graphics, GKS is a landmark which should have a significant effect on the way graphics is done over the next few years. Its structure is sufficiently novel that GKS programs will look different from the graphics programs using existing de facto standard packages. Whether it will be a success will depend to some extent on the way people adapt to this way of working but also to how good a job has been done in defining the standard. A bad standard or one that provides no significant advantage over current practice is probably worse than no standard at all.

References:

1. 'Methodology in Computer Graphics' edited by R A Guedj and H A Tucker. North Holland 1979.

2. 'Methodology of Interaction' edited by R A Guedj, P J W ten Hagen, F R A Hopgood, H A Tucker and D A Duce. North Holland 1980.

3. 'The Semantics of Graphic Input Devices' by V L Wallace. Computer Graphics, Vol 10 No 1 Spring 1976, pp 62-65.

4. 'Status Report of the Graphics Standards Planning Committee of ACM/SIGGRAPH'. Computer Graphics, Vol 11 No 3, Fall 1979.

5. 'Status Report of the Graphics Standards Planning Committee of ACM/SIGGRAPH'. Computer Graphics, Vol 13 No 3, Fall 1979.

6. 'Special Issue: Graphics Standards' ACM Computing Surveys Vol 10 No 4 December 1978.

7. 'Towards Compatible Graphics Standards' by P J W ten Hagen and F R A Hopgood. Mathematisch Centrum Amsterdam 17/79 February 1979.