Future develoments in graphics and workstations

F.Hopgood
D.Duce
Informatics Division Rutherford Appleton Laboratory (United Kingdom)

1. INTRODUCTION

The application of Computer Aided Design has been fragmented so far due to the lack of standards at the hardware and basic software level. The most impressive products have been turn-key systems using custom-built hardware with large software suites developed over a number of years. Such systems have often been difficult to modify and maintain. The very nature of such systems is that they are expensive to produce, have a limited market and, consequently, are expensive.

Hardware and software advances over the last few years point to a change in this environment. The trend is towards hardware and software compatibility from the computer suppliers allowing software suppliers to target their offerings at a wider range of products. This produces a competitive market and the downward trend in hardware costs gives the possibility for systems of much lower cost and, consequently, opens up the market to a larger customer base.

This paper will concentrate on the developments in single user workstations and graphics standards which should provide a firm base for this new environment.

2. HISTORY OF SINGLE USER WORKSTATIONS

The computer world was dominated in the 1960's by a batch mode of working. The 1970's saw a move towards interactive computing with either mainframes or mini computers with many users each getting a share of the resources available. The high cost of memory, disc storage and central processor power made it uneconomic to provide interactive computing in any other way. The appearance of powerful multi-user systems such as the VAX 11/780 and PRIME 750 near the end of the 1970's allowed better quality interaction and raised user's expectations.

The continuing decrease in cost of main and disc memory with an increase in the processing power of standard chips made it possible at the beginning of the 1980's for significant processing power to be made available in the user's terminal or workstation on his desk. This, together with the introduction of high quality bit map displays, has made it viable for the major part of the user's computational requirements to be provided by his local dedicated workstation at a cost which was initially of the order of £20K but has been decreasing over the last few years.

However, this would not in itself be a major breakthrough if the user was isolated from other users so that the benefits of the large mainframe, with a single database shared by a set of users, were lost. The advent of high speed local area networks allowed the possibility for the workstations to be connected together and share resources.

In summary, the hardware for a single user workstation is:

(1) High Speed Processor
(2) Large Main Memory
(3) Access to large 'local' disc storage
(4) High Resolution Display
(5) High Speed Local Area Network Connection
(6) Access to servers required by the application domain.

The first commercial company to offer a general purpose system that approached these requirements was the Three Rivers Corporation in Pittsburgh with the PERQ in 1979. This was followed about a year later by Apollo in the autumn of 1980 and SUN much later in 1982.

By 1983, a review conducted at Rutherford Appleton Laboratory (HAL) had 120 companies indicating that they had offerings in this area. Of these, about 55 already had products that fitted the overall criteria and 41 of these were based on the UNIX operating system [1]

The leading contenders at that time were:

(1) ICL PERQ2: ICL had developed a follow-up to the initial PERQ with Three Rivers. UNIX had been made available under a joint development with HAL.

(2) APOLLO DOMAIN: Apollo had developed a proprietary token ring with a fully distributed AEGIS operating system. It had the first fully integrated colour system. With a good FORTRAN environment it was able to attract software vendors to port software to the system and for product suppliers to oem Apollo hardware for their own systems.

(3) SUN: the company's strength was in the staff recruited from Berkeley which gave it the best 68000-based UNIX environment. Its distributed file store architecture was very crude and window management facilities, as with Apollo, were much more limited than on the PERQ.

Significant advances have occurred over the last three years and it is worthwhile updating the figures above in order to get some better starting point for extrapolation of the future. Of the systems above, the PERQ and SUN were selling around £20K with a comparable Apollo system being significantly more expensive.

3. WORKSTATION SCENE AT THE START OF 1987

3.1 Introduction

The major casualty in the 3-year period has been the Three Rivers/ICL PERQ. The PERQ2 had been largely a reworking of the original PERQ1. For most of 1984 and 1985, the two companies worked on a PERQ3 which was significantly lower in cost and higher in performance than the PERQ2. Due to impressive gains by the other two companies in the market, ICL decided to abandon the PERQ3 and base its few products on SUN hardware despite the PERQ3 being superior to the SUN3 in a direct comparison. The lack of a range of products indicated to ICL that they would have difficulty competing in the market.

The trends in SUN and Apollo have been similar. Apollo's main changes in philosophy have been to bring them more into line with other companies accepting the need to have UNIX as an operating system and non-prorietary local area network connections.

The next section will indicate the changes at SUN over the period as an example of what has happened in general. Most of the statements are equally applicable to the other workstation vendors.

3.2 SUN 1983-1987

SUN1 systems were first shipped in May 1982. The major step forward in 1983 was the launch of the SUN2 system range. These were based on the MC68010 processor with Multibus for connecting discs and magnetic tapes. The main features were:

(1) UNIX Berkeley 4.2 Operating System - an industry standard in the academic community and elsewhere.

(2) Range of Systems these started as low-cost disc-less nodes requiring support over the network. The main offering was a stand-alone black and white system supplemented by both colour systems and file servers. This compatible range allowed SUN to cover a much wider market than a single system.

(3) Memory the systems could have between 1Mbyte and 4Mbytes with a 16Mbyte virtual memory system.

(4) Display Resolution a maximum of 1152x900 on the monchrome systems and 640x480 on the colour ones.

(5) Window Management a layered window management system Sun Windows allowing systems designers to integrate applications at a lower level if performance requirements made it necessary.

(6) Ethernet a 10Mbits/sec ethernet provided file access and paging to disc-less workstations over the local area network.

The SUN2 systems were powerful workstations but demanding applications still required more performance than was available. Consequently, the trend over the last few years has not been to reduce the price but to increase the performance. Also, there were considerable gaps in software available on these systems and so the other thrust has been to make the basic software more comprehensive and encourage vendors to port their software to the range.

The major advances over the last three years have been:

(1) Improved Performance the early 68010 10MHz systems have been replaced by 68020 16MHz systems with a range of accelerators to increase performance in specific areas - floating point, graphics etc.

(2) Enhanced Memory memory sizes were increased to 16Mbytes with a 256Mbyte virtual address space.

(3) Larger Discs the earlier 35 and 70Mbyte disc capacities were increased to 380Mbytes and the ability to put multiple drives on the one workstation.

(4) VMEbus the earlier Multibus was replaced by the industry standard VMEbus with dual-ports allowing the slave processors access to both memory and the main cpu as well as allowing disc and tape connections.

(5) Network File System an efficient and reliable protocol which has become an industry standard for access to a remote file server from a workstation.

(6) Price: the price per system has varied very little, but the performance per system has increased significantly.

The following gives some idea of the performance improvements from the early SUN2 systems to the latest SUM (using the VAX 11/750 as a normalising factor):

	SUN2	SUN3
Integer Operations	0.82	3.22
Floating Point Operations	0.41	8.08

The new SUN3 systems are substantial computation engines. Improvements in SUN specific functions:

	SUN2	SUN3
Linpack - double precision	1	16
Whetstones - double precision	1	7
Whetstones - single precision	1	17
Paging	1	2
Graphics	1	18

Over the last few years there has been a considerable increase in overall performance resulting in workstations that can handle quite a wide range of applications in the design area. The price has remained relatively stable with system prices ranging from just over £10K for a disc-less node to £40K for a powerful colour workstation.

4. THE FUTURE

4.1 Introduction

What are the key advances likely over the next few years? To a large extent this has to be guess-work but some trends are beginning to appear. In 1987, SUN are likely to overtake Apollo as the main workstation supplier. However, competition will be intense from the mainframe manufacturers (IBM, DEC etc) and the PC world (Apple, Atari, IBM) together with efforts from the established terminal suppliers (Hewlett Packard, Tektronix) to get a larger share of the market.

4.2 Processors

Improvements in cpu performance have mainly been achieved by improvements in the microprocessor industry. The 68000 to 68010 to 68020 has been the source of most of the improvement together with associated chips for floating point and other operations.

This technology is likely to be used to drive the basic price of a system down while retaining the current performance levels. Already the PC suppliers are enhancing their systems with the more recent chip sets. The Atari ST, for example, is likely to be extended eventually to be a dual-processor 68020-based system running UNIX while keeping the price below £3K. Such a system would open up a wide range of new application areas for the workstation market.

There is some indication that the top-end processor power is unlikely to meet the industry's needs if it continues to rely on microprocessor enhancements. For this reason there has been considerable interest in Reduced Instruction Set Computers (RISC). The mainframe and mini systems such as the IBM 370 and DEC VAX have rich instruction sets where high performance is achieved by sophisticated compilation and architectural aids such as memory interleaving, I/O processors, cache memories etc. The RISC approach is to design a system with a much reduced instruction set. The more complex operations of the existing order codes are performed as a series of instructions. A major advantage is that the complete cpu can be produced in VLSI with large sets of registers to allow fast context switching.

The current market consists of companies such as PYRAMID, RIDGE, IBM 6150, }1P9000, all of which have brought out RISC architecture systems. There are signs that other workstation manufacturers are also moving in this direction. MIPS, a California based company, has developed a processor board which is being made available to other workstation suppliers. Speed improvements of between 5 and 8 over the SUN3 are anticipated.

A more novel development is to improve performance by multiple processor systems using one of the many novel architectures under development. An example in the UK is the INMOS transputer which is being used in a variety of environments and architectures with as many as 60 or 80 processors in a single system.

One of the advantages for software suppliers over the last few years has been the dominance of the 68000 microprocessor range in the workstation market. This has considerably eased the porting of software to a number of workstations. Unless some de facto standard appears, it is likely that system architectures are likely to diverge which will put more pressure on suppliers having a compatible software environment on different underlying architectures. This problem will apply to the vendors such as SUN who may well have conventional microprocessor based products at one end of the range and proprietary RISC-based architectures at the other.

4.3 Servers

With NFS becoming the de facto standard for file access over an ethernet, the industry has the ability to define a range of servers to interwork with workstations conforming to this standard. Already SUN, APOLLO, WHITECHAPEL, DEC, IBM and Atari systems can be linked using NFS.

The initial offerings have been fileservers with Pyramid, Could, Sequent and others offering competition to SUN in this area. Some of these systems, which are also targeted at the small mainframe market, have substantial cpu performance and can also be used as additional computation power.

For demanding applications requiring significant cpu power, many companies are offering compute servers to work in this environment. Most use vector or parallel architectures to achieve the performance. For example, Active Memory Technology is offering a reworked version of the ICL 32x32 DAP which consists of an array of single processors. Alliant have a multi processor system which is being integrated into the Apollo range. A recent survey indicated that over 30 companies were offering products in this general area.

A requirement in the CAD area is fast database servers, Apart from the ICL CAFS system, very few have appeared as yet with a solid market base. This is likely to become an important area in the future.

The server philosophy does allow sharing of expensive I/O facilities. The most common printer in use at the moment is the A4 laser printer from companies such as Apple. Excellent output quality has made these the basis for nearly all the desktop publishing markets. Most of the initial systems used the Canon engine but competition is appearing and it is likely that the cost of a quality laser printer will be driven down to the £1K or £2K price.

A major problem is still the area of large size output devices. The major vendor is still Versatek with its expensive electrostatic systems. Although a number of companies are competing with Versatek for this market, there has not been a significant reduction in price and these are still expensive particularly if colour is required.

4.4 Graphical Hardware

A variety of special purpose hardware is being developed and included in the graphics workstations and displays coming on the market. At the lowest level these enhancements provide hardware assistance for the basic graphics primitives (area fill, arc generation, image processing). At the intermediate level they provide geometric operations such as 3D transformations and clipping. At the highest level, they provide storage for graphical objects and the ability to render them with some realism.

An example of a workstation with geometry hardware is the Silicon Graphics IRIS. The company started in 1983 using the Geometry Engine developed by Jim Clarke. Each IRIS 2000 system has a pipeline array of 10 or 12 geometry engines each performing a specific geometric operation (matrix transformation for rotate, translate and scale; clipping; perspective). Up to 65,000 transformations per second can be achieved on the standard system.

A more recently developed chip, the Geometry Accelerator, provides buffering and floating point conversion which allows light source shading and hidden surface removal to be achieved in real time for reasonably complex objects. Hardware and firmware support is also provided for Gouraud shading, depth cueing and Z-buffering.

Both SUN and Apollo have recently introduced their own graphics enhancements. SUN have a Graphics Processor which can be added to their main colour workstation. It performs similar operations to the Silicon Graphics Geometry Engine. The SUN Graphics Buffer is similar to the Geometry Accelerator above and is dedicated to speeding up hidden surface removal operations.

Many of the performance boosters depend on the graphics being produced in a certain way. The 3-D object has a particular representation and the operations performed on it are defined in a particular way. To ensure that these systems are compatible with the software being used to drive them, there is a need to standardise the graphics software which is the subject of the second half of this paper.

5. GRAPHICS STANDARDS - AN INTRODUCTION

After more than 10 years of effort by many people from a number of countries, graphics has its first international standard in GKS, the Graphical Kernel System. Rather than being the end of the road it is just the beginning. GKS is the main building block of a set of inter-related standards that are due to appear over the next few years with the aim of producing a comprehensive set to cover the graphics area. A description of the history and basic concepts of GKS are given in [2]

Standards provide a mechanism for formally specifying the exchange of information across an interface. GKS concentrates on standardising the interface between the application and the graphics system together with the interface between the graphics system and a workstation (which in GKS terms can be thought of as an intelligent device capable of controlling a variety of input devices, a display surface and providing local picture manipulation and storage of a simple nature). Figure 1 gives a representation of GKS and its interfaces.

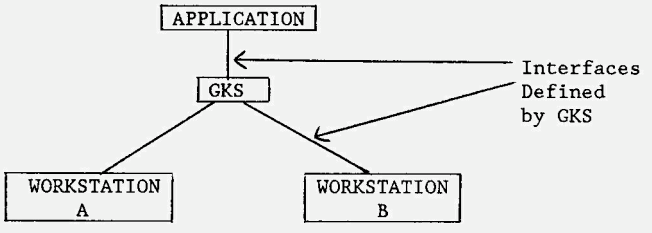

Figure 1: Interfaces defined by GKS

Both GKS interfaces are defined functionally with no specification of how they should be realised in terms of a language interface to the application or a protocol between GKS and an intelligent device. Consequently, it is feasible for a variety of systems to arise all conforming to the GKS definition but having totally different characteristics.

The future standards activities are partially aimed at making the existing interfaces more tightly specified but also introducing other interfaces and providing greater functionality. Figure 2 shows how the current standards activities fit together.

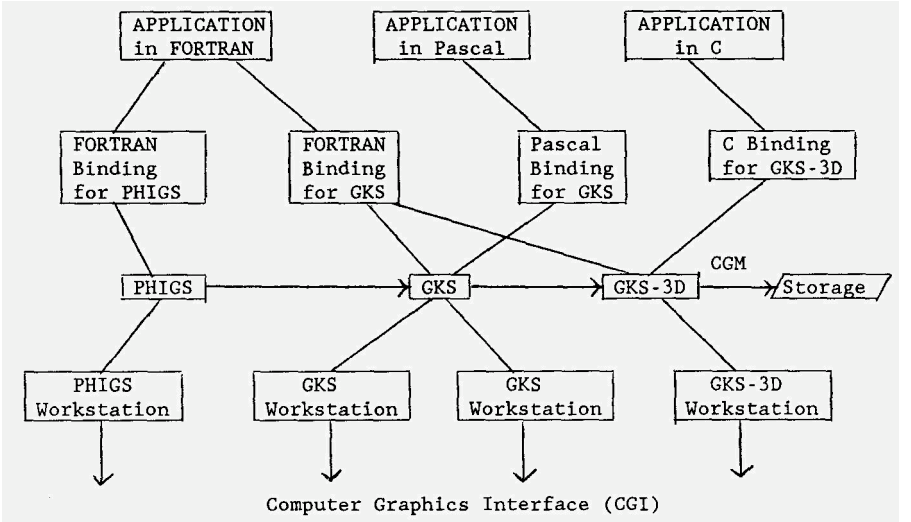

Figure 2: Standards

Not all the interfaces are shown in Figure 2. For example, there will be a range of language bindings to all the graphics standards rather than the limited number shown. The main thrusts of the standardisation activities following GKS have been:

1. To provide language bindings to be used with GKS.

2. To provide protocols for long-term storage of graphical information (CGI - Computer Graphics Metafile).

3. To provide a standard interface to graphics devices (CGI - Computer Graphics Interface).

4. To increase the functionality of the standards by extending GKS to 3D and also providing a modelling facility (PHIGS).

The rapid increase in the number of activities has caused individuals involved to concentrate on one or other of the standards activities. As a result, there is a danger that the various activities will drift apart. To avoid this, the aim is to standardise a Reference Model of how the various standards interact.

6. ISO TERMINOLOGY

In any activity that involves communication between people, a few terms are used so regularly that they become an extension to the vocabulary. The standards area is no exception and it is important that a reader is aware of the various stages that a standards document goes through within ISO before becoming an international standard. These are:

1. WORKITEM an official project with agreed scope and goals and timescales. When an area has been identified for standardisation, a proposal for a project is prepared. There is a ballot on the proposal within the appropriate Technical Committee (TC) and, if successful, the workitem is assigned to a particular Subcommittee (SC), who manage the project and in turn assign it to a particular Working Group (WG) who carry out the technical work. (The Working Group for computer graphics is WG2 within SC21 (Open Systems) within TC97 (Information Processing Systems) designated TC97/SC21/WG2.)

2. DRAFT PROPOSAL (DP) the Working Group produces successive working drafts of the standard until the document is sufficiently mature that it can be submitted to the Subcommittee for registration as a DP. The DP is then circulated within the SC for technical review and ballot. Comments submitted with the votes are addressed and resolution of them is sought. If sufficient agreement is reached the document proceeds to the next stage. If not, or if the document has undergone substantial change, then it has to be circulated for a further DP ballot.

3. DRAFT INTERNATIONAL STANDARD (DIS) when sufficient agreement is reached on the DP document, the revised document is registered as a DIS. The publication of a DIS should indicate that technical agreement has been reached. The document is then circulated within the TC for editorial review and DIS ballot. Comments submitted with the votes are addressed and resolution sought. Any remaining problems at this stage can cause another DIS ballot, but normally the document proceeds to the next stage.

4. INTERNATIONAL STANDARD (IS): the DIS revised in the light of comments received with the DIS ballot becomes the Final Text. A final ballot within ISO Council ensures that all ISO members are satisfied that ISO procedures have been followed in the production of the standard and that the document is suitable for publication. The IS is then published. The document will be reviewed 5 years after publication at which time it may be endorsed, revised or abandoned.

The voting process in ISO is by letter ballot and takes many months each time. Consequently progress is slow and getting to a full international standard is a long and time-consuming activity requiring considerable stamina.

A question immediately arises as to when a new standard proposal is sufficiently well developed to warrant either implementing or using. In theory, all technical change should be complete by the DIS stage and this is, therefore, a reasonable time to start using a standard. However, the move from DIS to IS inevitably includes some technical revisions if only

to remove ambiguities. Consequently, time should be allowed for updating any programs written at the DIS stage when the IS finally arrives. In the case of GKS, minor changes did occur between the DIS and IS stages. As a result, most of the books currently on the market have minor errors. The only one known to be up-to-date is [3].

Pilot implementations of standards often occur at the DP stage. This is right and proper as such implementations often point out problems in implementation. A danger with such products is that there is a tendency to massage the DP version to produce the DIS one with a consequent loss of efficiency and elegance. Choosing a particular implementation of the standard should at least give some thought to the history of the product. Probably the best product would be a new DIS one where the company had tested out algorithms and techniques on an earlier DP prototype.

The dates for GKS were as follows:

Workitem	Spring 1981
Draft Proposal	February 1982
Draft International Standard	June 1983
International Standard	August 1985

Note the long elapsed time between Workitem and International Standard. Predicted dates for future standards should be judged against this schedule to decide how likely the progress will be made to keep the predicted dates.

7. GKS

The Graphical Kernel System - CKS is formally defined in the standard document itself [4]. A more detailed introduction and primer is given in Hopgood et al [3] while a full and comprehensive treatment with many examples is given in Enderle et al [5]. The latter book is in the process of being updated to agree with the International Standard and current FORTRAN binding. It should appear later in 1987. A computer graphics text book based on GKS that can be recommended is Ream and Baker [6]. Readers should be warned that it has a few errors as it uses an earlier version of the FORTRAN binding.

Rather than give a full description of GKS here, we will concentrate on those areas which are significantly different from previously accepted packages or are major features of CRS and the related standards.

7.1 Dimensionality

GKS is a two-dimensional graphical system and provides no support for three dimensions. The major reason for this was that it was realised that the standardisation of a 3D system would take significantly longer than a 2D-only system. There was an urgent need for a standard and large parts of industry had no interest in 3D. Consequently, the right approach was to move quickly to the definition of a 2D standard with the intention of defining a 3D standard above the 2D system at a later date. The extension of CRS to 3D will be described later.

7.2 Primitives

The six basic output primitives are polyline, polymarker, fill area, text, cell array and generalised drawing primitive (COP).

Previous packages including the CSPC CORE system were based on the concept of Current Position (CP). The usual line drawing primitive in such packages is to generate a line from CP to a specified point followed by updating CP to be the specified point.

CRS, on the other hand, defines output of this type by specifying a sequence of points and the polyline output primitive draws a set of lines between the sequence of points. One advantage of this approach is that aspects such as dotted or broken apply to the complete polyline rather than individual line segments, a more natural view when drawing curves depicted as polylines.

The polymarker primitive is similar to the polyline primitive but marks the sequence of points with a specified symbol rather than connecting the points with lines.

The text primitive in GKS provides considerable flexibility in defining the quality of the text, its size and orientation, the origin etc. Figure 3, where the asterisk defines the text origin, indicates the types of text available in CRS. It also supports the text path being in any of the major directions providing support for those languages not writing from left to right.

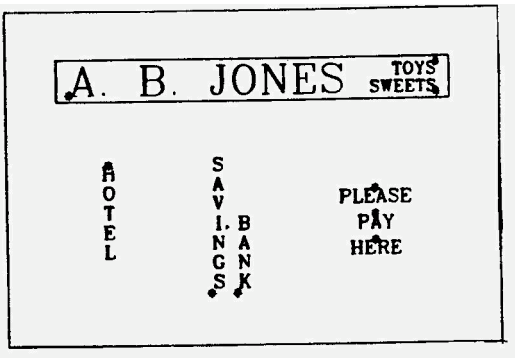

Figure 3

The fill area primitive is defined in terms of a set of points which specify a closed curve. The primitive fills the enclosed area with a solid colour or allows it to be filled by a specified pattern or hatch style.

The cell array primitive is specifically aimed at the image processing community where the cell array defines the colour or grey level to be associated with individual elements of a rectangular array.

Finally, CDP defines a controlled method of adding more exotic primitives. Particular implementations are free to add to the basic primitive set by specifying particular CDP types as producing higher level shapes such as circles, ellipses etc.

The appearance of primitives on a display is determined by the aspects of the primitives. For example, the aspects of a polyline are: linetype, linewidth scale factor and polyline colour index. Linetype may be solid, dashed, dotted, dashed-dotted or other implementation dependent possibilities. Linewidth scale factor defines linewidth as a function of a standard linewidth for the output device. The polyline colour index points to a colour description. The method of setting the values of aspects will be described shortly.

7.3 Coordinate Systems and Workstations

GKS has introduced the concept of an abstract workstation to hide the peculiarities of device hardware. A workstation consists of zero or one display surfaces and zero or more input devices. GKS assumes that applications will frequently want to use more than one workstation simultaneously. For example, an operator may be interacting with a design through a refresh display, whilst taking copies of completed parts of the design on a plotter.

Coordinate data in the parameters of an output primitive are specified in world coordinates (WC), a Cartesian coordinate system. Transformation to the coordinate system of the display device is accomplished in two stages; firstly, world coordinates are transformed to an intermediate coordinate system called normalised device coordinates (NDC) by a window to viewport mapping termed a normalization transformation, then a second window viewport mapping, called the workstation transformation transforms these coordinates to device coordinates (DC). The aspect ratios of window and viewport may differ in the normalization transformation, but the workstation transformation maps the workstation window to the largest possible region of the workstation viewport with the same aspect ratio.

A major difference between GKS and other earlier systems is that it provides multiple normalization transformations all defined at the same time. Coordinates of primitives are transformed by the currently selected normalization transformation. Figure 4 shows three different objects (duck, tree and house) defined in different world coordinate systems. Three different normalization transformations can be defined which map these world coordinates onto specific areas of the NDC space.

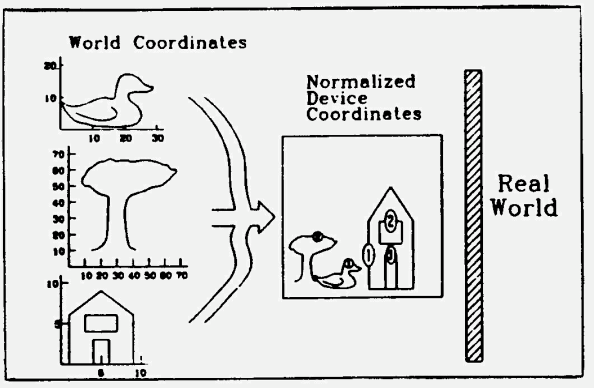

Figure 4

This leads to a different style of programming with the normalization transformations, rather like declarations, being defined at the head of the program and not changed. Earlier systems tended to mix calls of primitives with changes in coordinate system.

The workstation transformation may be set differently for different workstations, thus allowing different parts of the virtual picture to be displayed on different workstations.

The workstation can be regarded as a camera pointing at some part of the NDC space. In Figure 5, the first workstation is pointing at the tree while the second points at the house.

The boundary of the window of the currently selected normalization transformation serves as a clipping rectangle against which output primitives may be clipped. There is also a compulsory clip to the boundary of the window of the workstation transformation.

The Viewing Pipeline in GKS is given in Figure 6.

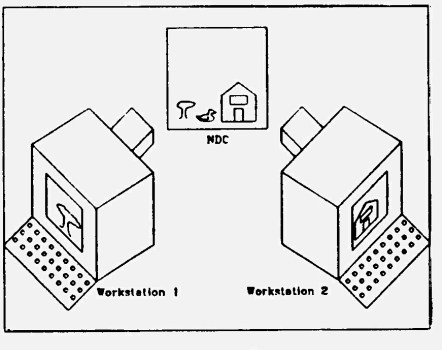

Figure 5

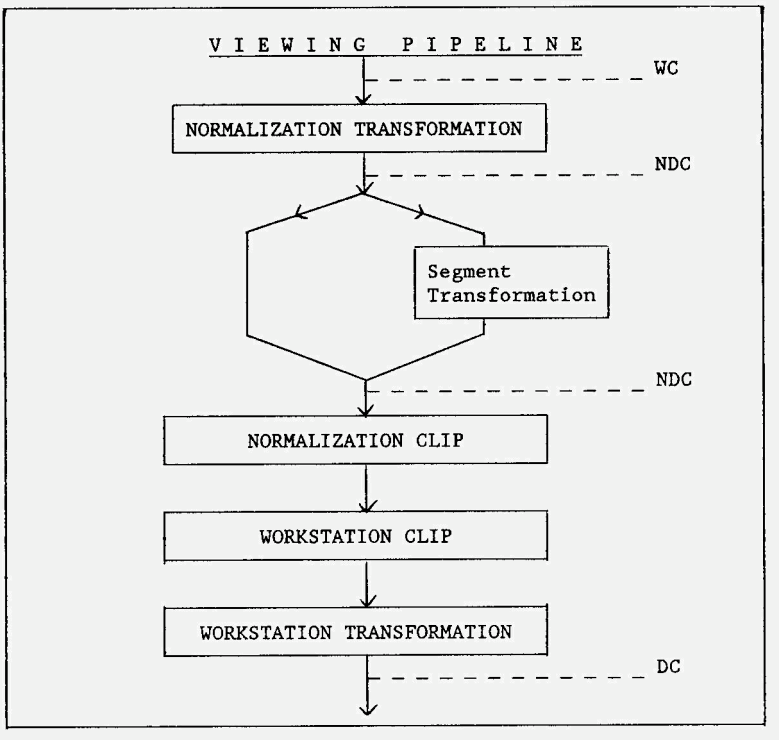

Figure 6

7.4 Attributes

As noted above, the appearance of a primitive displayed on a workstation is determined by its parameters and by additional data termed aspects. The values of aspects are determined by attributes. Aspects fall into two categories: geometric and non-geometric. Geometric aspects control the shape or size of a primitive, for example the height of a text primitive. Each geometric aspect is controlled by a single geometric attribute. Geometric attributes are specified in world coordinates, are set modally and are subject to the normalization and workstation transformations. A primitive has the same geometric aspect values on all the workstations on which it is displayed. Only the text and fill area primitives in fact have geometric aspects.

Non-geometric aspects control facets of the appearance of a primitive which are not related to its shape or size, for example the linetype (solid, dotted etc) with which a polyline is displayed. The values of non-geometric aspects may be controlled in one of two ways: bundled specification - there is one attribute per output primitive which controls the values of all the non-geometric aspects; individual specification -there is one attribute for each non-geometric aspect.

The polyline primitive will be used as an example. In the bundled scheme, the values of all the polyline aspects are determined by the value of the polyline index. A polyline index defines a position in a table, the polyline bundle table. Each entry in this table is termed a bundle and specifies values for all the non-geometric aspects of a polyline. The bundle corresponding to a particular polyline index is termed the representation of the index. When a polyline is created, the current value of the polyline index is bound to it and cannot subsequently be changed.

In Figure 4, the picture in NDC space has the various polylines making up the picture identified by their polyline index value. The duck has the same polyline index value (1) as the outline of the house. The tree has the same index value (2) as the upper window. This effectively defines that in the virtual world the duck and house outline will look the same but they can be differentiated from the tree, upper window and also door (polyline index value 3).

The important point about bundle tables is that each workstation has its own bundle tables and so a polyline in the virtual picture may be displayed with different representations (ie different bundles corresponding to the same polyline index) on each of the workstations on which it is displayed.

For example, the duck, which has a polyline index attribute equal to 1, can be represented on workstation 1 by green solid lines while workstation 2, if it is a monochrome device has the ability to define it as a thick dotted line.

This is a powerful tool for achieving application program portability between different workstation environments. If carefully constructed, moving a program to a different environment will merely mean defining new representations for the different indices used in the picture, to employ in the best way possible, the characteristics of the workstations in the new environment.

In the individual scheme the values of the polyline aspects will be the same on all the workstations on which the polyline is displayed and each workstation must do the best it can to display the polyline with the requested aspect values.

The bundled scheme is important then when it is necessary to ensure that primitives with different attributes can be differentiated on different workstations; whilst the individual scheme is important when primitives with specific attributes are to be represented on each workstation as closely as possible to the specification.

7.5 Input

One of the areas which was considerably refined during the GKS review process was the model of graphical input. Just as output was defined in terms of device independent primitives and attributes, the aim was to specify a set of virtual input devices on to which real input devices could be mapped.

All input devices in GKS were formalised as having a measure and a trigger. The measure describes the type of input value returned by the device (position, value, text etc) while the trigger causes the measure value to be returned to the application program in certain styles of input.

Six logical input devices are defined in GKS each delivering measures as follows:

LOCATOR : a position in world coordinates and the associated normalization transformation number used to convert back from device coordinates via NDC to world coordinates.

STROKE : similar to LOCATOR but delivering a complete sequence of world coordinate positions.

VALUATOR : a real number.

CHOICE : an integer representing a selection from a set of choices.

PICK : the name of a selected segment and an identifier indicating which set of primitives in the segment has been picked.

STRING : a string of characters.

A GKS implementation supporting input must provide a simulation of each logical input type to the operator. They need not all be on one workstation if he has several workstations under his control.

The three operating modes in which GKS input devices may be set to provide input are:

REQUEST : rather like a FORTRAN READ. A request is made by the application program for a measure of the specified device to be returned. CKS will wait until the operator has set the measure to the desired value and activated the trigger. The measure value is returned to the application program which then continues executing.

SAMPLE : the measure is continually updated and *GKS* returns the current value of the measure whenever the application program requests it. For SAMPLE input, the trigger is not required.

EVENT : a number of input devices may be active together. Each time the trigger for a particular device is activated, the current measure value is added to a single queue of input events for all the devices in use in EVENT mode.

The application program can interrogate the queue acting on the events that have taken place. It is possible to couple more than one input device to the same trigger so that multiple events at the same time are possible.

The local installation decides how the real input devices are modelled to provide the necessary logical input devices.

A major difference from many interactive graphics systems is that all logical input devices can be used in all three operating modes.

7.6 Segments

The segment model in GKS is less innovative than some parts of the standard. Associated with each workstation is a segment store in which segments consisting of sets of GKS commands can be stored. Functions exist to create, delete, rename and manipulate segments. Associated with each segment are a set of attributes which control visibility, highlighting, priority ordering for output when segments are overlayed and detectability from a pick device. It is also possible to transform segments such that the picture defined by the segment can be scaled, moved, rotated etc.

As well as the segment store associated with the workstation, there is also a workstation independent segment storage (WISS) which is used as a central library. Segments can be moved from WISS to a workstation. A macro facility is also provided so that segments in WISS can be inserted into other segments.

7.7 Levels

Rather than insist that all facilities in GKS are supported by every implementation, GKS is defined as a set of levels on two orthogonal axes:

o : simple output
1 : output including segments
2 : full output including all the facilities for inserting segments from WISS into the current segment
a : no input
b : REQUEST input only
c : all forms of input

There are, therefore, 9 levels in total with Oa the simplest and 2c the most comprehensive.

7.8 Summary

GKS is the first international standard. Its major limitation is that it is a 2D standard. In compensation, its attribute and input models are much better than those previously used. Its coordinate systems provide a greater level of flexibility and device independence than most earlier systems.

7.9 GKS Implementations

The two early implementations of GKS which are still widely available are GKSGRAL which derives from the version implemented at the Technical University of Darmstadt and the joint ICL-Rutherford Appleton Laboratory implementation which is widely used in the UK university environment.

Since these early implementations, there have been a number of additional ones falling into three classes:

(1) Device Manufacturers these implementations are oriented towards the hardware of the particular manufacturer although the host package is usually portable.

(2) Host Manufacturers implementations by a mainframe manufacturer for his range of systems hopefully with a good coverage of popular devices.

(3) Software Houses these are products aimed at being sold to other organisations for their products (either device or host).

A list of implementations (to Level 2b unless indicated) with FORTRAN as the main binding (unless indicated) is:

(1) Rutherford Appleton (RAL)/ICL (Level lb going to 2b)
(2) GTS-GRAL (Level 2c)
(3) NOVA Graphics, Austin
(4) CWI Amsterdam (Level 2c)
(5) Tektronix
(6) CEEGEN, Los Gatos
(7) Whitechapel, UK
(8) Prior Data Sciences, Ottawa (C Binding)
(9) Precision Visuals, Boulder
(10) Dataplotting Services Inc
(11) RAXTEK
(12) Visual Engineering, San Jose
(13) Advanced Technology Centre, Culver City (Level 2c)
(14) TEMPLATE
(15) UNIRAS, Mass
(16) DEC (Level Ob)
(17) Data General
(18) IBM/Graphic Software Systems
(19) Infolytica, Montreal (Level mb)
(20) AED-GKS (Level 2c)
(21) CMC, Bombay
(22) Sysgraph, Vienna
(23) System Simulation Ltd, London (Level 1a)
(24) XGKS, Hungary

8. GKS-3D

8.1 Introduction

This seeks to extend GKS to 3D by adding various capabilities, as can be seen from the scope given in the Draft Proposal:

(a) the definition and the display of 3D graphical primitives;

(b) mechanisms to control viewing transformations and associated parameters;

(c) mechanisms to control the appearance of primitives including optional support for hidden line and/or hidden surface elimination but excluding light source, shading and shadow computation;

(d) mechanisms to obtain 3D input.

The aim is to specify the system in such a way that existing (2D) GKS programs would run without any modifications and that the general style of capabilities provided would match those included in GKS.

To achieve compatibility between GKS and GKS-3D, existing GKS 2D functions are still provided in GKS-3D but, conceptually, have a Z=D coordinate added to every position. As a result, existing GKS output sits on the Z=D plane. The default viewing transformations provided will produce a parallel projection on to the same part of the workstation display screen as if the output had come from a GKS 2D system.

8.2 Output Primitives

GKS-3D has seven output primitives. Six of these correspond to the GKS primitives, the seventh is fill area set which displays a set of polygonal areas (this is particularly convenient for specifying areas with holes or disjoint areas that are to be treated as a single entity). This primitive was added because it was felt that in a 3D environment rendering may be needed across a set of areas.

Text, fill area, fill area set and cell array are planar primitives in arbitrary planes. Planar primitives have an obverse and a reverse side (and zero thickness), determined uniquely from the parameters and aspects of the primitives. Characters of the text primitive and patterns of the area primitives are generated on the obverse. Viewing the reverse displays a mirror image of the obverse.

GKS-3D provides both 3D and 2D functions to generate instances of the primitives. The 3D functions accept 3D coordinate data whereas the 2D functions only accept 2D data and generate primitives in the Z=D plane (in world coordinate space).

The appearance of fill area set primitives is determined by the fill area aspects and by a new set of aspects controlling the appearance of the edges of the primitive. An edge bundle table has been introduced with corresponding bundled and individual attributes.

8.3 Viewing

The Viewing Pipeline in CRS-3D is given in Figure 7. Similar to CRS, a transformation exists to change the user defined world coordinates into a consistent Normalised Device Coordinates for internal use within CRS-3D. The next operation in the pipeline is to view the NDC picture. For most workstations viewing consists of projecting the 3D image on to a 2D projection plane. Functions are provided to assist with the definition of this viewing operation. There is a change in coordinates from Normalised Device Coordinates (NDC3) to Viewing Coordinates by defining a View Reference Point and a set of axes associated with it. The intention is that this point has some relationship to the object to be viewed and makes the setting up of the projection transformation that much easier.

Once the Viewing Coordinates are established, Front and Back Planes are defined which specify the limits of the object to be viewed. A Projection Reference Point can be specified and a Projection Plane which allows the object to be viewed by projecting it onto the projection plane. The View Window specifies that part of the projection plane to be output to the workstation. Both parallel and perspective projections are provided.

For some workstations, capable of providing 3D geometric transformations and for genuine 3D devices, the viewing operation specified by the functions provided may not be appropriate. Therefore, it is possible for applications to construct their own viewing pipeline or ignore parts of it.

8.4 Multiple Views and Hidden Surface Calculations

Each primitive in CRS-3D has a View Index associated with it which defines which viewing transformation is to apply to it on a particular workstation.

It was believed that, unlike CRS, there is a need for more than one view to be available at a time on a workstation. This would, for example, allow titles to be output using a parallel projection while a 3D object to which the titles are associated is output using a perspective transformation. It can also be used to provide plan and elevation views of the same object.

Support for Hidden Line end Hidden Surface calculations is provided at the workstation level. Associated with primitives is an attribute defining which method of rendering is to be used on the workstation. The workstation can be asked to render or not and it has flexibility in how it does the rendering. Consequently, a variety of workstations can choose the most appropriate methods depending on their hardware characteristics.

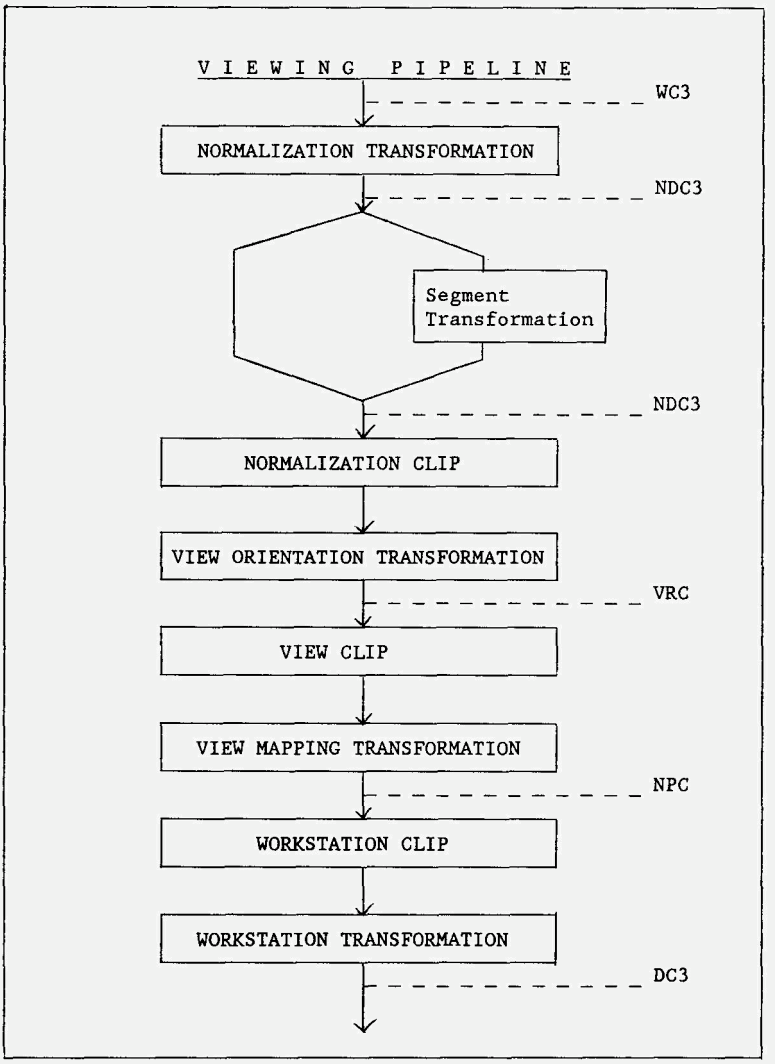
V I E W I N G P I P E L I N E
WC3
NORMALIZATION TRANSFORMATION
NDC3
Segment
Transformation
NDC3
NORMALIZATION CLIP
VIEW ORIENTATION TRANSFORMATION
VRC
VIEW CLIP
VIEW MAPPING TRANSFORMATION
NPC
WORKSTATION CLIP
WORKSTATION TRANSFORMATION
DC3

Figure 7

9. PHIGS

9.1. Introduction

A major problem with GKS-3D is that it does not cater for the most sophisticated end of the device market nor does it adequately handle the application where there is a need to make rapid changes to the complex hierarchical pictures often found in some areas of CAD.

The problem is not so much with the primitive definitions, attribute models or viewing transformations but with the segmentation facility which does not allow the application database and graphical picture structure to be closely integrated. In fact, the one-level segmentation facility in GKS forces any hierarchical filestore to be built on top within the application program. However, by doing this, it makes it almost impossible to use hierarchical segmentation facilities if they are available in the device.

The solution, in terms of standardisation activities, has been to extend GKS in two different directions. The first is GKS-3D and the second is the Programmers Hierarchical Interactive Graphics System (PHIGS).

9.2 Main Features

The primitives and attributes in PHIGS closely follow those in GKS-2D and GKS-3D. The viewing models in GKS-3D and PHIGS are also very close. The one major difference between PHIGS and GKS is that in PHIGS the creation and display of a picture are very explicitly independent phases. (It is worth noting that PHIGS and GKS are not as far apart as it might at first appear, though this point cannot be developed here.)

At the heart of PUGS is a single structure store. A structure consists of a number of structure elements which can be both graphical and non-graphical. Thus it is possible to keep application data associated with graphics in the same database. A more likely approach is that the Application data in the PHIGS structure store will be pointers to the relevant entries in the application database. The PHIGS structure store replaces the GKS segmentation facility, though the two do not occur at the same point in the viewing pipeline.

Structures are not displayed on workstations until they are posted to workstations. Posting a structure to a workstation causes the structure to be traversed,, generating graphical output for display on the workstation. Structure elements representing application data are ignored by traversal, Once a structure has been posted, changes made to the structure are reflected on the workstation until the structure is unposted. Unposting a structure does not cause it to be removed from the central structure store.

Particular features of the structure facility are:

(1) Hierarchy structures can call other substructures and the same substructure may be called more than once from a higher level. Thus a car may need only a single wheel substructure which is called four times.

(2) Modelling Coordinates: structure elements contain positional information in modelling coordinates. Each structure has a global and local modelling transformation which are concatenated to produce the transformation to be applied to points to turn the modelling coordinates into the coordinates to be passed to the viewing pipeline.

(3) Inheritance: substructures inherit attributes from the calling structure. Thus, the global modelling transformation is the one passed in by the calling structure. Similarly, attributes such as colour can be passed to the substructure.

On completion of traversing a structure, control reverts to the higher structure that called it and the attributes are reset to those in force on entry to the substructure. Thus the substructure can have no effect on the calling structure.

(4) Editing: labels can be placed in structures and there is a structure element pointer. Consequently, it is possible to move around a structure and edit it after initial creation. This is unlike GKS segments which cannot be changed once the segment is created.

For high quality displays, it is possible for structure traversal to be done by the hardware in the workstation allowing fast graphical movement of complex pictures in 3 dimensions.

9.3 Implementations of GKS-3D and PHIGS

Only two implementations of GKS-3D are commercially available at the time of writing (as far as we know):

(1) GKSGRAL-30 this is available from GTS-GRAL in Darmstadt. It is upward compatible with the company's GKSCRAL 20 system. The implementation does include optional modules to support hidden line/hidden surface calculations. Only a FORTRAN binding is available. The system runs on a range of hardware from PCs to host mainframes. The current implementation is to Level 2b. A number of drivers are available for standard terminals.

(2) CMC: CMC Ltd of Bombay, India have a GKS-3D implementation which in the UK is being sold through device manufacturers. The product has a FORTRAN binding which predates the ISO OP and the company has indicated that it will change the FORTRAN binding to agree with the ISO one when it becomes stable.

With the current early stage of PHIGS in the standardisation process, it is too early to talk about commercially available implementations. However, a pilot implementation has been produced by IBM and Rensselaer Polytechnic Institute. Details of this implementation are given in [8]

A number of manufacturers have indicated that their existing computer graphics software is PHIGS compatible (for example, Apollo and Megatek). We have been unable to assess these products to see how closely they coincide with the PHIGS ISO working draft. As the Apollo one is based on their current graphics system which has significant differences from PHIGS, it is unlikely that full compatibility can be achieved either with the current working draft or the future standard.

10. GRAPHICS INTERCHANGE

10.1 Introduction

GKS provides a standard for graphics in two dimensions (both input and output). The philosophy in GKS is that the operations requested by the application are for almost immediate action. The segmentation facility provides an on-line method of storage of transient graphical information but is not designed for long-term storage between sessions. Once the workstation is closed, the segment store ceases to exist.

GKS recognised the need for storage of graphical information between sessions and initially included within it a GKS Metafile facility as part of the standard which allowed an audit trail of GKS commands (used to create and manipulate pictures) to be stored and later retrieved and executed.

Once it became clear that there was likely to be more than one graphics standard at the functional level and all would have a need for long term storage and retrieval, it was decided to separate out the metafile function as a separate standard. That standard activity is the Computer Graphics Metafile (CGM).

The final GKS standard retains a set of functions for reading and writing metafiles. The intention is that these functions could be used to read and write CGM metafiles. However, there is also a need to provide more specific metafile facilities specifically for the GKS environment. GKS provides an Annex to the standard where a protocol is defined for communication in the GKS environment. This protocol will provide an audit trail as described above. The Annex is not an intrinsic part of the standard but, if present, will allow communication between GKS systems or long-term storage and auditing within a GKS system. It has greater functionality than CGM in the area of segmentation. The GKS metafile, for example, can be used to store a set of segments. A later use of GKS could read in these predefined segments. On the other hand, CGM is much more a facility for picture storage.

The GKS metafile looks very much like a workstation. Once the special workstation defined as a metafile is opened, any graphical commands obeyed are stored in the metafile. This continues until the metafile is closed. This similarity between a workstation and metafile implies that there is a close relationship between the protocol used to define the metafile and that required to define the interface between CRS and the virtual device. As shown in Figure 2, the standards activity to provide an interface to the graphical device is CGI, the Computer Graphics Interface. In this section, we will show the inter-relationship between CGM and CGI.

10.2 Computer Graphics Metafile CGM

The Computer Graphics Metafile (CGM) is a file of more or less device independent graphics orders. It provides a standard for:

(1) Retaining and transporting graphical data defined as pictures.
(2) A data interface for graphics packages.
(3) A picture transfer mechanism between different devices, installations and systems.

CGM is defined as being compatible with GKS but allows a wider range of functionality so that it can be used for interchange of graphical information in a wider context than just GKS. A key element in the philosophy is that the process creating the information in the CGM can be separated in time and space from the process using it. Thus CCM could be used to generate a magnetic tape to be read at a remote installation many weeks later using a different type of graphics system from the one that generated it.

The elements making up the CGM are broadly split into seven classes:

(1) Description Elements: these elements specify the version of the CGM used in defining the file and information concerning the capabilities of the process needed to read the CGM.

(2) Control Elements : these elements define the size and orientation of the space in which the CGM is defined.

(3) Picture Descriptor Elements: the CGM provides more flexibility in some areas than GKS. For example, line width in GKS is specified by giving the width of the line as a factor of the standard line width on the specified device. The CGM allows as an alternative the width to be specified in virtual device coordinates. The descriptor elements declare the modes in use for this particular CGM.

(4) Graphical Elements: these describe the visual components of the picture being transferred. They include the GKS output primitives as a subset but also include elements for the efficient transfer of circles and arcs.

(5) Attribute Elements: these specify the attributes of graphical elements and are equivalent to the GKS attribute model.

(6) Escape Elements: these describe device or system dependent elements where no constraint is placed on the contents.

(7) External Elements: these elements are used to include relevant messages and application data not directly related to the graphical image of the picture.

The CGM is effectively transporting a virtual picture and, consequently, defines all picture elements in Virtual Device Coordinates which are closely coupled to Normalised Device Coordinates.

The CGM description includes the coding of how the information is formatted. The CGM standard document now consists of four parts. The first contains the functional specification of any conforming metafile. The other three parts contain specifications of three methods of encoding, each with its own particular goal:

(1) Character Encoding this is intended for use where it is important to minimise the size of the metafile; where necessary, this is regarded as more important than processing speed. This encoding makes it suitable for transmission through 'ASCII' networks.

(2) Binary Encoding this encoding aims to minimise the processor effort required to generate and/or interpret the metafile. It is therefore highly suitable for storage and retrieval of graphical data within a computer system.

(3) Clear Text Encoding this encoding is aimed at the requirement of having a metafile that can be read and edited by people. It is also very safe to transport, even between systems with different native character sets.

In addition, the document allows private encodings as long as they conform to the Functional Description and general rules of conformance in Part 1. An example of this would be a binary encoding using word lengths or representations other than those specified in Part 3.

10.3 Computer Graphics Interface CGI

There is a close affinity between the CGM standard and the Computer Graphics Interface (CGI) standard. This is not surprising as both are seeking to describe pictures by a linear sequence of commands.

The CGI defines the interface between the device independent and device dependent parts of a graphics package. Unlike the CGM standard which defines the output from a graphics package for transmission or storage, the CGI standard must handle both output and input and it is assumed that the device is on-line and capable of supporting dynamic interactive graphics.

The CGI has to support a whole range of terminals from simple plotters to high powered interactive terminals. To do this in a sensible way, physical devices are required to support a minimum set of functions such as line drawing and may support, but are not required to, a richer set of functions including functions such as arc and circle generation.

The device driver on the device side of the CGI will generate the necessary device codes for the required functions and will either emulate the non-required functions in terms of the required ones if the device does not support them or uses the device functions if it does.

The CGI supports segmentation and input while the CGM does not. Consequently, the set of elements defined in the CGM is augmented by input elements and the picture description elements including segmentation. The CGI, however, is not a superset of the CGM as the Descriptor Elements are not provided in the CGI. Instead, interrogation of the device by the graphics system establishes the attributes of a device prior to a session starting.

At this point in time, there is also a major difference in that the CGI contains a set of commands specific to the control of raster devices. As well as cell array, CGI also includes a more device dependent primitive called pixel array which performs functions on the actual physical pixels of the device. The BITBLT operation is provided for moving, copying and constructing bit maps.

The CGI includes s very large and complex set of functions (the current document is 536 pages long). Many of the functions are inappropriate for many devices, however it is not a straightforward matter to identify simple subsets of the functions and group them in sensible ways. Early working drafts used the idea of option sets to provide access to particular facilities (for example colour and segmention). This approach has now been changed because of the difficulty of agreeing the content of the option sets. The approach taken in the DP document is based on the idea of a constituency profile These profiles define sets of functions and their precise capabilities for particular classes of CGI users. It is intended that the CGI standard itself will define a, number of such profiles related to the needs of GKS, and other profiles will be subject to Registration.

10.4 ICES

10.4.1 Background

ICES stands for 'Initial Graphics Exchange Specification' . It is a standard for the transfer of CAD/CAM information. IGES is thus a standard for the exchange of application data in contrast to CGM which is a standard for the exchange of graphical data. The two standards thus address very different requirements.

ICES was developed initially as a format for the transmission of representations of 2D engineering drawings between dissimilar CAD/CAM systems, motivated by a very real practical problem being faced by many large companies. However, since the development of ICES started, there have been important changes in the way engineering parts are represented by CAD/CAM systems. 2D draughting systems have given way to 3D wireframe systems, including the capability for defining complex free-form surfaces. The emerging generation of CAD/CAM systems are based on solid modellers. The concern now is with complete product models, which will contain not only the geometry of the part, but also a great deal of other information, for example manufacturing information. IGES is evolving into a standard for product definition data exchange.

The development of ICES started in the US in the late 1970's under the auspices of the National Bureau of Standards. The first version was based on a data exchange format developed by Boeing in the late 1960's which was known to work. The format was finalised in 1980 and became an ANSI standard in 1981. It is incorporated in ANSI Y14.26M 'Digital Representation for Communication of Product Definition Data'. This is ICES Version 1. Versions 2 (1982) and 3 (1985) incorporate various improvements, for example better facilities for representing surfaces. Version 4, under development, is looking at incorporating solid modelling information.

10.4.2 Overview of ICES

The minimum requirement for a standard for the transmission of product definition data is that it should be able to transmit geometric data, annotation and organisational information. ICES treats a product definition as a file of entities, each of which is represented in an application-independent format, to and from *which the* representations of

specific CAD/CAM systems can be mapped. Three types of entity can be transmitted; geometrical (lines, arcs, spline curves etc used to compose the model), annotation (dimensions, construction lines, arrows and textual notes such as appear on an engineering drawing) and structural.

Because of its origins, ICES (and its successors) is based on an 80-column card image format. A binary format has been introduced more recently which reduces the data storage requirements of an ICES file by at least 50%.

An ICES file consists of S sections.

(1) User comments. This section is meant to provide a human readable introduction to the file.

(2) Global section. This section contains global information necessary for interpreting the file, for example the units used in the file, the characters representing certain delimiter functions.

(3) Directory Entry section. This is effectively a dictionary of all the entities transmitted in the file.

(4) Parameter Data section. This section contains the data defining each entity. Pointers link each directory entry with its corresponding parameter data and vice versa. Pointers are in fact card sequence numbers!

(5) Terminator card.

An additional Binary information section precedes a binary ICES file. This contains information concerning precision and the length of each section following.

10.4.3 The Future of IGES

Initially GES was slow to gain acceptance, largely because CAD/CAM vendors were not committed to the concept. Vendors were more keen to see IGES files translated to their system than the other way round, consequently preprocessor-translators were often in advance of the corresponding postprocessors. However, increasing pressure from customers has lead to major improvements over the last two years. There are some incompatibility problems which remain, notably in the area of free-form curves and surfaces.

IGES is being used by a number of large organisations and certainly goes a good part of the way to solving the problem. There is no better alternative at this point in time.

The developers of ICES are currently developing a proposal called PDES (Product Data Exchange Specification) which builds on the IGES experience and will be much more forward looking. Two notable influences on the design of PDES are XBF (Experimental Boundary File, developed by the international organisation CAM-I, for the transmission of solid modelling data) and PDDI (Product Definition Data Interface developed under the US Air Force ICAM program).

The French company Aerospatiale have developed a format called SET (Systeme d'Echange et de Transfer) which it is claimed gives 5 or 6-fold improved translation times over ICES.

ISO technical committee 184 now has a working group in this area developing a standard called STEP (Standard for Transfer and Exchange of Product Model Data) which is attempting to merge the above work into one common approach. This work is unlikely to result in an international standard before 1990.

11. REGISTRATION

During the evolution of all these standards it has become obvious that it is not possible to standardise everything at once. In particular, there are a number of graphical elements that can be found in a bewildering number of varieties. An example would be the set of all 'useful' marker types. Any one person could probably think of a hundred or more: a specialist could probably think up a thousand in just his own field.

Rather than delay the standards in progress by trying to get agreement on extensive lists of such elements for each standard in turn, the documents now just refer to a single registration mechanism and mandate only a very small number of such elements. The registration mechanism is being set up to deal with the standardisation of the following elements initially.

- Generalised Drawing Primitives (GDPs)
- Escapes
- Line types
- Marker types
- Hatch styles
- Text font usage
- Prompt/Echo types
- Error messages

This registration mechanism will, by default, provide for the extension of each relevant graphics standard in each of these areas. Thus for an extra marker type, it will define the appearance of the marker, allocate a marker type number for CKS, do the same for the CGM (including each of the bindings) and so on for each standard. In some cases, the element will not be appropriate (for example, Prompt/Echo types are not relevant to the CGM).

By this means, it is expected that the requirement for extensions to the standards in these areas will be met - without having to update each standard - and also that all the standards will stay in step with minimum effort and confusion.

The US National Bureau of Standards has been approved as the Registration Authority for the Register of Graphical Items. However, the precise procedures to be followed by the Registration Authority are still being defined.

12. STATUS OF GRAPHICS STANDARDS

Only GKS has so far reached full International Standard status. The following table gives our best guess as to when each project will reach its next stage in the ISO pipeline. By their very nature, such dates are approximate as they depend on the results of ISO ballots. Any decision to have a second DP or DIS Ballot will increase the timescales. As each DP Ballot is out for voting for three months and DIS Ballots for six months, any second iteration will cause a significant delay.

Once a project has reached DIS stage, it can be regarded as reasonably complete and unlikely to change in significant ways. At that stage, products could be based on the standard proposal with only minor retrofitting being necessary. On the other hand, if a proposal has only reached the DP stage, it is quite likely that significant changes will appear before it reaches DIS status.

From the dates below, it can be seen that GKS and the FORTRAN and Pascal Bindings are almost complete. GKS-3D and CGM are getting to a stage where major changes will not occur in the future. It should be stressed that both CGI and PHIGS are at a very early stage of standardisation and will inevitably have changes made to them before they become Draft International Standards.

Project	Expected Progress		
	DP	DIS	IS
GKS	-	-	-
GKS Language Bindings			
FORTRAN	-	Jul 86	Jun 87
Pascal	-	Nov 86	Jun 87
ADA	-	Oct 86	Jun 87
C	Nov 86	Oct 87	Dec 88
GKS-3D	-	Dec 86	1988
GKS-3D Language Bindings			
FORTRAN	-	Feb 87	Mar 88
Pascal	Mar 88	Dec 88	Jan 90
ADA	Mar 88	Dec 88	Jan 90
C	Mar 88	Dec 88	Jan 90
PHIGS	Dec 86	Dec 87	Dec 88
PHIGS Language Bindings			
FORTRAN	(dependent on PHIGS dates)		
Pascal			
ADA			
CGM Functional Description	-	-	Jan 87
Character Encoding	-	-	Jan 87
Binary Encoding	-	-	Jan 87
Clear Text Encoding	-	-	Jan 87
CGI	Dec 86	May 88	Dec 89

The dates given for expected progress are the dates at which the DP and DIS ballots start. A minimum of 15 months delay is likely between the start of the DIS ballot and the publication of the IS.

13. WINDOW MANAGEMENT

13.1 Introduction

With the advent of single user systems with bit map displays, it is possible for a single operator to work effectively on a number of separate tasks allowing the window manager to provide individual windows which appear to the operator like separate devices. For a single application, it is possible for the window manager to multiplex several workstations on the same bit map display. The graphics system needs the window manager to define areas of the screen as virtual devices. The window manager needs the graphics system to define icons, window borders etc. A key problem is how should the two relate. This section will pose some problems and indicate some possible solutions. For further discussion of these issues see [9]

13.2 History

The earliest mention of a window management system is in Alan Kay's 1969 thesis [10] at the University of Utah where he states that 'the display for FLEX was a large virtual screen on which displays may be tacked rather like notices on a notice board'. The major input device associated with window management systems is the mouse which Doug Engelbart designed at Stanford Research Institute in 1963. These dates show that the concepts were available early on but progress was restricted to a few research laboratories due to the high cost of hardware.

The first realisation of a window management system was at Xerox PARC where Larry Tesler, Dan Ingalls and Alan Kay produced a system for SMALLTALK which introduced the overlapping window paradigm. Windows on the screen could be put on top of one another like papers on a desk. Operations were provided to rearrange the order of the window. In the early SMALLTALK system, you could only interact with the top exposed window.

Later systems at Xerox PARC (several produced by Warren Teitelman) introduced and experimented with features such as scrolling, window borders, pop-up menus, icons etc. the overlapping window paradigm was questioned and tiled window managers were tried whereby individual windows were carefully positioned in a regular order on the screen with no overlapping.

The appearance of the Three Rivers PERQ meant that there were commercially available single user systems with high resolution bit map displays capable of supporting a window management system at low cost. APOLLO and SUN systems soon followed and the appearance of the MACINTOSH at the low end of the market made window management systems and the associated paradigm widely available.

13.3 Model

A general model of window management systems is given in Figure 8.

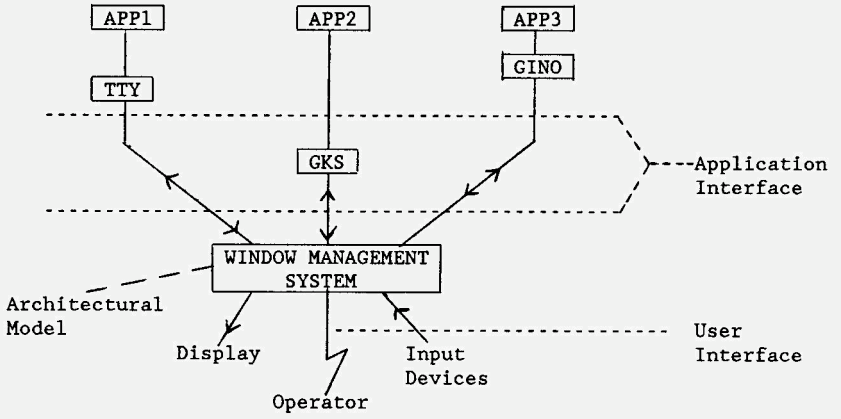

Figure 8

Three major areas for discussion are:

(1) Application Interfaces at what level should the application interface be placed. At the two extremes you could have the graphics system producing pixel changes in windows or being incorporated as part of the window management system. Almost certainly, the interface will be somewhere between those two extremes.

(2) Architectural Model both the user and the application need to understand the architectural model of the window management system. As can be seen from the history, various facilities have been tried without them being put together in a sensible reference model.

(3) User Interface many of the existing systems appear the same on the surface but frequently have different philosophies once the surface is scratched. Some consistency in design principles, while still leaving the ability to do commercial tailoring, is essential.

In the commercial offerings so far there has been almost no commonality in any of these three areas. Several UK manufacturers have been working towards a common Application Interface (CSI: Client Server Interface) with RAL [11].

13.4 NeWS

A major move in the window management area has been SUN's decision to push their new base window management facility, NeWS, as a de facto standard in much the same way as they have been successful with NFS.

NeWS, standing for Network/extensible Window System, is a platform upon which a variety of window management systems could be built. It allows programs running on systems anywhere in the local network to use windows on a workstation. It is based on a similar philosophy to the UK CSI. Clients wishing to access resources request a Server to supply the resource. The Clients are the window-based applications. The resources controlled by the Server are the bit map screen and input devices.

Across the Client Server Interface, applications need to send low level graphical commands and receive input from devices. In order that good quality echoing of input is achieved in the application's window that is appropriate to the application, it is necessary for the application to control this echoing. However, the application is at a distance, conceptually, from the display. The solution in NeWS is that the application down-line loads a program to the server which is entered when input occurs.

To control the CSI, a protocol is required which is efficient, can handle input and output, and is a programming language. Rather than develop a new language, SUN have extended Adobe's POSTSCRIPT language to allow input as well as output. POSTSCRIPT is currently the de facto standard for running laserprinters such as the one on the Macintosh. This approach has allowed SUN to capitalise on the acceptance already received by POSTSCRIPT in the workstation environment.

It will be interesting to see whether NeWS achieves the popularity currently enjoyed by NFS.

14. CONCLUSIONS

The future for CAD appears very bright at the moment. The local area network environment of workstations connected to servers appears to becoming the correct architecture for a locally distributed system. The hardware costs will decrease significantly so that substantial functions can be performed with good interactive capabilities in the workstation. The appearance of servers allows sharing of resources and data in the local environment.

Interface standards such as CGM and CGI in the ISO arena and de facto standards such as NeWS, NFS and POSTSCRIPT mean that the application can be separated from the vagaries of the hardware.

The emerging graphics standards will allow a considerable part of the interactive control process to be moved from the application to the graphics system thus reducing the size of application programs, making the product more competitive.

REFERENCES

[1] F.R.A.Hopgood, C.Prosser, 'Single User Workstations', Computer Graphics Forum, 2, 1983, pp219-223.

[2] P.R.Bono, J.L.Encarnacao, F.R.A.Hopgood and P.J.W.ten Hagen. 'GKS - the First Graphics Standard', IEEE Computer Graphics and Applications, Vol 2, No 5, July 1982.

[3] F.R.A.Hopgood, D.A.Duce, J.R.Gallop and D.C.Sutcliffe. 'Introduction to the Graphical Kernel System GKS', Academic Press (2nd edition) 1986.

[4] 'Information processing systems - Computer graphics - Graphical Kernel System (GKS) functional description', ISO 7942, ISO Central Secretariat, Geneva, August 1985.

[5] G.Enderle, K.Kansy and G.Pfaff. 'Computer Graphics Programming, GKS - The Graphics Standard'. Springer-Verlag 1984.

[6] D.Hearn and M.P.Baker. 'Computer Graphics'. Prentice-Hall 1986.

[7] 'Digital Representation for Communication of Product Definition Data'. ANSI Y14.26M - 1981.

[8] S.S.Abi-Ezzi and A.J.Bunshaft. 'An Implementer's View of PHIGS', *IEEE Computer* Graphics and Applications, Vol 6 No 2, Feb 1986.

[9] F.R.A.Hopgood, D.A.Duce, E.V.C.Fielding, K.Robinson and A.S.Williams, 'Methodology of Window Management', Springer Verlag, 1985.

[10] A.C.Kay, 'The Reactive Engine', PhD Thesis, University of Utah, 1970.

[11] A.S.Williams, 'An Architecture for User Interface R&D', *IEEE* Computer Graphics and Applications, July 1986.