CERN 83-07
Data Handling Division
21 July 1983

ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

A BACKGROUND TO COMPUTER GRAPHICS

D.R. Myers

GENEVA
1983

CERN — Service d'information scientifique – RD/597 – 2000 – juillet 1983

ABSTRACT

Computer graphics is introduced to those who are new to the subject. After briefly covering the possible applications in high-energy particle physics, the report surveys the different types of display hardware available, the more important terms and concepts used, and the facilities for interaction with the computer made possible by bit-mapped graphics workstations. A brief description is given of the international standard Graphical Kernel System (GKS) as well as a three-dimensional graphics software package, PIONS, which is being developed at CERN.

CONTENTS

CHAPTER 1

WHY GRAPHICS ?

1.1 INTRODUCTION

The material for this report was originally assembled for a talk given during April 1983 to the software section of the LEP DELPHI Collaboration. I have since been asked to repeat the talk and so, as the information may be of wider interest, I was 'persuaded' by various people to write something down on paper. The intended readers are those physicists who:

i) need to decide how to make use of Computer Graphics in new experiments;

ii) intend to write programs using Computer Graphics;

iii) are users of graphics equipment and think they should know something about its possibilities and limitations.

The information presented here represents a personal choice of background material. I would strongly recommend those with a serious interest to consult Foley and van Dam [1]. In particular, anyone intending to write code for an interactive computer application should look through the first six chapters of this work. One final point to dispel any possible misconceptions. This report is not intended to be a review of existing graphics software in high-energy physics (HEP) and does not cover application programs, such as histogramming packages, etc. A guide to such information will be found in Rafelski [2].

1.2 APPLICATIONS FOR GRAPHICS IN HIGH-ENERGY PHYSICS

In the early days of computers their range of applications was imagined to be rather narrow. With the increase in performance and drop in price due to the advent of the microprocessor, this range has expanded enormously. The same effect is now being seen in computer graphics. The hardware costs are becoming so low that it is now economically justifiable to make use of graphics displays in a whole range of new applications. Moreover, advances in display architecture are making it possible to use completely new techniques for humans to interact with computers. In particular, I refer readers to the work done at Xerox Palo Alto Research Center [3,4], which has led to the user interfaces provided for SMALLTALK, the Xerox Star, Perq, and the new 'Lisa' personal computer from Apple.

A display system is suited to a particular application depending on certain characteristics. Some of these may be enumerated as follows:

i) colour or monochrome

ii) resolution of displayable points (i.e. 500 * 500 to 4000 * 4000)

iii) selective erase of information on the screen

iv) drawing speed

v) Vector or Raster screen

vi) 2 or 3 dimensions.

Categorizing applications is difficult, but below they are divided into two for convenience. It is implicitly assumed that any package making use of graphics to communicate with a user will allow interaction. Output-only packages should be restricted to making 'hard' copies onto paper or film. [Note that the term 'resolution' is used fairly indiscriminately in this report to indicate the number of addressable points on the display surface. It should, more precisely, be used to define the number of visually distinguishable points].

1.2.1 Low-resolution 2D Graphics -

Until recently most computer graphics came under this heading. Typical HEP applications are:

i) Numerical Data Presentation
For example, the display of graphs, histograms, and scatter plots. This application is typically output only, although graphical interaction would be useful for choosing scales, arranging the layout, etc., before making hard copies on paper or film.

A very important aspect of this application is the production of diagrams, and transparencies for publications and talks. This is an area which has been sadly neglected, at least at CERN. However, the coming of Personal Workstations with bit-mapped graphics should significantly change the situation. It requires the integration of both text and graphics editors, graphics input, and a high quality output device.

ii) Experiment Control Systems
This application includes the display of menu trees, schematic diagrams, and numerical data presentation.

1.2.2 High-resolution 3D Graphics -

This category implies that the screen should have at least 1000 * 1000 addressable points, and that the application program should be able to specify the graphics in 3D. The 3D to 2D projection may be performed either in hardware or by a graphics package. A list of possible applications should include:

i) Detector Design
One may imagine here a program providing the interactive capability to place pieces of a detector in space in a way similar to a mechanical computer-aided design (CAD) system. By feeding the result into a Monte Carlo event-generation program, such as GEANT [5], it should be possible to test rapidly the efficiency of the design.

ii) Pattern Recognition
This is especially useful for detectors producing 3D coordinates, such as multiwire proportional chambers (MWPCs) with charge division and time projection chambers (TPCs). An interactive 3D display system lets the programmer really see the situations in which pattern recognition algorithms fail.

iii) Detector Check-out
This includes looking at calibration data, such as field maps, and also at test data to see how the complete

detector functions when all the individual pieces are assembled together. Other points to include under this heading might be the checking out of triggers (both hardware and algorithms) and event backgrounds.

iv) Event Reconstruction and Scanning
It may be imagined that with more than a few thousand events it would not be possible to make use of an interactive graphics system for this purpose. However, after the triggering algorithms have been used to reduce the event sample, those events left are the ones of greatest interest, or with which the software has the most difficulty. Hence, being able to display them is likely to be essential. It must be acknowledged that there is a law of diminishing returns in making programs increasingly complex to cover all special cases.

v) Patch-up and Analysis
The comment from the previous point applies here also. Having arrived at an event sample of particular interest it may be very useful to be able to re-fit interactively individual tracks, for example to check whether inclusion or exclusion of particular points makes significant changes to the track's parameters. Interactive viewing programs which provide kinematic fitting may also help to debug algorithms designed to search for particular features, such as jets.

1.3 DYNAMICS AND COLOUR

For most, if not all, of the second set of applications, a display system which allows dynamic manipulation of a 3D picture is very advantageous. Experience gained with the Merlin graphics facility for the experiments UA1 and UA2 shows that whilst the image is rotating the brain perceives it as having depth. Thus one can immediately see whether points lie in the plane of a particular track. In this respect there is a major qualitative difference if a display system generates 2D projections at 10 to 20 Hz, or only once every few seconds. In the first case it is possible to disentangle complicated high-multiplicity events; in the second case it is not.

Colour is a feature which is just starting to become available with 3D dynamic displays. It provides a fourth dimension not available in monochrome and enables the display of non-geometric information. Thus tracks may be colour coded to show their energy, momentum, or fitted mass. Similarly, the energy deposited in calorimeters may be indicated by their colour, or colour may be used to display confidence levels, etc.

CHAPTER 2

DISPLAY HARDWARE

To appreciate the limitations and possibilities of computer graphics it is necessary to have some acquaintance with the available technology. The aim of this chapter is to mention briefly the different display types and their 'ball-park' price ranges. It must be stressed that prices change rapidly, and so those quoted here are only intended to give an idea of the cost at the time of writing.

2.1 PICTURE STORAGE

It may not be obvious to everyone that after a computer has drawn a picture on a display monitor the image, or a description of it, must be stored somewhere. This is commonly achieved in three ways.

2.1.1 Direct View Storage Tubes -

In the past the ubiquitous green Tektonix screen was almost synonymous with graphics. The direct view storage tube (DVST) stores the picture as charges on the inside surface of the screen. This gives rise to its major failing, which is that selective erasure of pieces of the picture is not possible. Also, it has only a single (low) intensity, and the adjustment may be somewhat critical. However the resolution is excellent, with up to 4000 * 4000 addressable points available.

2.1.2 Frame Memories -

Frame Memories are devices used for picture storage on raster display systems. They are required for bit-mapped workstation displays but, as explained later, do not by themselves provide all the required functionality. Thus, if a salesman says that his product has bit-mapped graphics, ask him if he knows what a 'RasterOp' is! (Raster Operations are explained in the section on workstations).

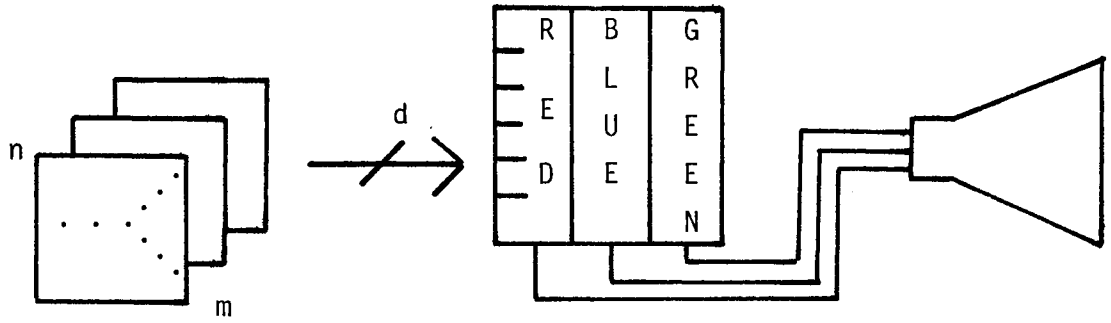


Fig. 1 - Frame Storage and Colour Look-Up Tables

A Frame Memory consists of d (for depth) logical planes of m * n bits. The m * n bits correspond to the addressable points on the screen. Thus for a monochrome picture with 1024 * 1024 points one requires 128 Kbytes of memory. By increasing the number of planes, or depth, of the memory it is possible to display variable intensities or colours. For example, with 3 planes each picture element, or 'pixel', corresponds to 3 bits, allowing 8 different colours. Of course, the logical arrangement just described does not necessarily correspond to any actual hardware implementation. Displays are available with up to 32 bit planes for image processing systems, but 4 to 8 planes are easily sufficient for most graphics applications.

This type of picture storage is becoming the dominant one now that memory prices have become so low. However, the technical standard required is not trivial. Consider that a 1024 * 1024 display with 4 planes needs 1/2 Mbyte of memory and an output bandwidth of 65 Mbits/s for a non-interlaced refresh rate of 60 Hz.

A technique associated with Frame Memories is the use of Colour Look-Up Tables. The aim here is to give greater freedom of choice of colour without prohibitive cost. The

integer formed from all the bits corresponding to a given pixel is used as the index into three look-up tables; one each for the red, green, and blue guns of a colour CRT monitor. If, for example, each look-up table is 4 bits wide, then the number of possible colours available will be 16 * 16 * 16 = 4096. The application will only be able to use 2 exp(d) colours simultaneously but, by loading the look-up tables differently, a much larger range of colours is available. A point to note is that on low-priced displays in exhibitions, colour look-up tables are often used to simulate movement. This gives them the appearance of having more functionality than is in fact the case. Buyers beware!

2.1.3 Display List Memories -

Display List Memories are essential for vector displays, but are also becoming available with the more expensive raster or even DVST systems (e.g. Tektronix 4114). Unlike the previous two picture storage methods, which memorize a facsimile of the image, the display list contains a description from which the picture may be generated. This allows a great deal more flexibility, because the display list may be used to regenerate the image with certain modifications; after performing a rotation, for example.

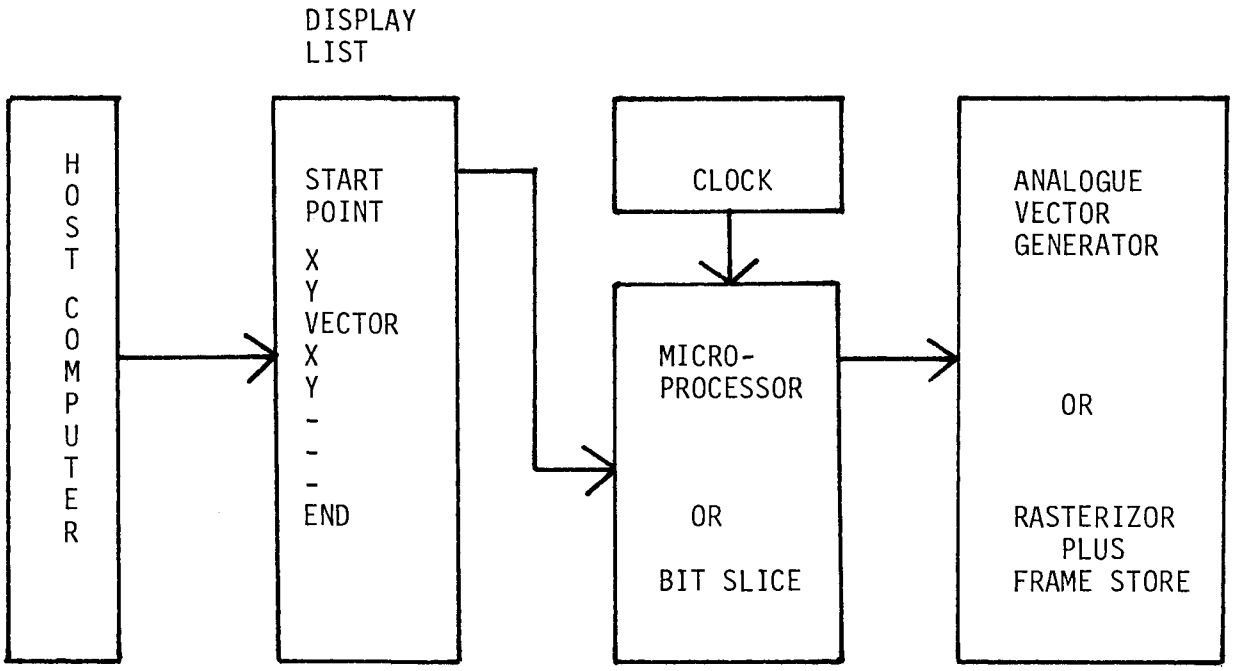


Fig. 2 - Display List Memory

Systems using display lists cover a wide range of sophistication. The simple ones just allow a sequence of commands such as: DRAW-2D-VECTOR, or DRAW-POINT, the end-points being specified as 10 or 12 bit integers. More expensive systems may include 3D with orthogonal and perspective projections and multi-level segmentation. The possibility to specify coordinates in floating point numbers is starting to appear and memory sizes vary from 64 Kbytes up to 1 Mbyte or more.

2.2 DISPLAY TYPES

The description of display types has been separated from that of picture storage because there is not necessarily a one-to-one correspondence. Below we consider complete display systems.

2.2.1 Storage Tube Displays -

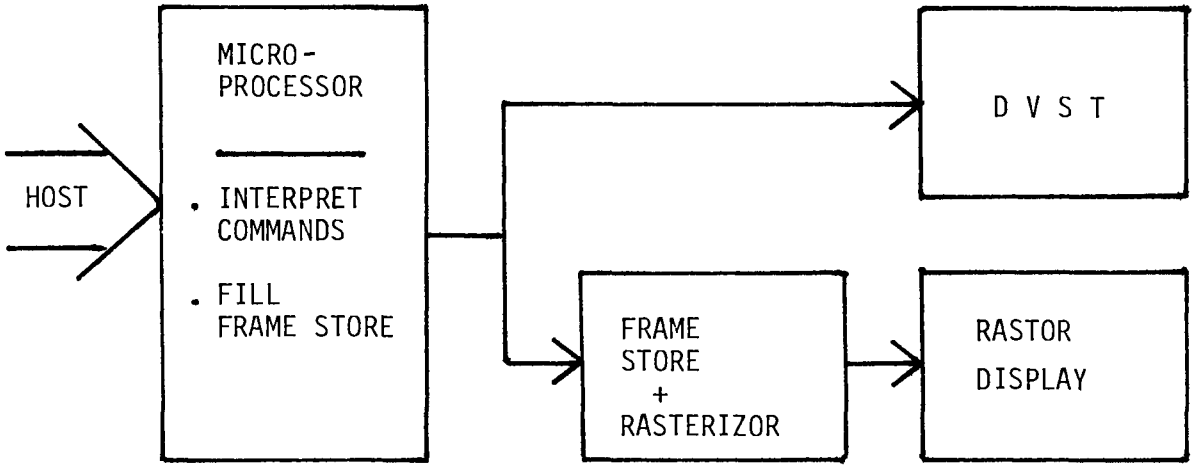


Fig. 3 - Simple Raster or DVST Architecture

The main features have already been described above, except to add that the displays are capable of drawing points, vectors, and hardware generated characters. DVST displays are usually designed to look like computer terminals (connected via RS232 lines). The drawing commands are interpreted by a simple processor and are specified as special character sequences preceded by an escape code. There is a de facto standard for these commands which has been set by Tektronix. The price range is of the order SF 25,000 -> 60,000 +.

Because of the serious limitations of the DVST, graphics terminals of this type are rapidly being superseded by raster equivalents (see below).

2.2.2 Refresh Vector Displays -

The basic elements of a Refresh Vector Display system are a display list memory, a graphics processor, and a fast XYZ CRT monitor. Vectors are drawn by applying voltage ramps to the X and Y axes; the Z axis corresponds to intensity. The processor runs on the line frequency or, more and more commonly, on an internal clock. At every 'tick' it interprets the display list and paints a picture on the CRT. If the list is too long to process in the time available then one of three things may happen, depending on the system. Either the end of the list is not shown; the processor misses a cycle, and so runs at 1/2 (1/3, 1/4, ...) speed; or the start of the next cycle is delayed. The last option clearly does not work with a processor synchronized to the line frequency. Some vector display systems make use of an intermediate display list containing partially processed information to help overcome this problem.

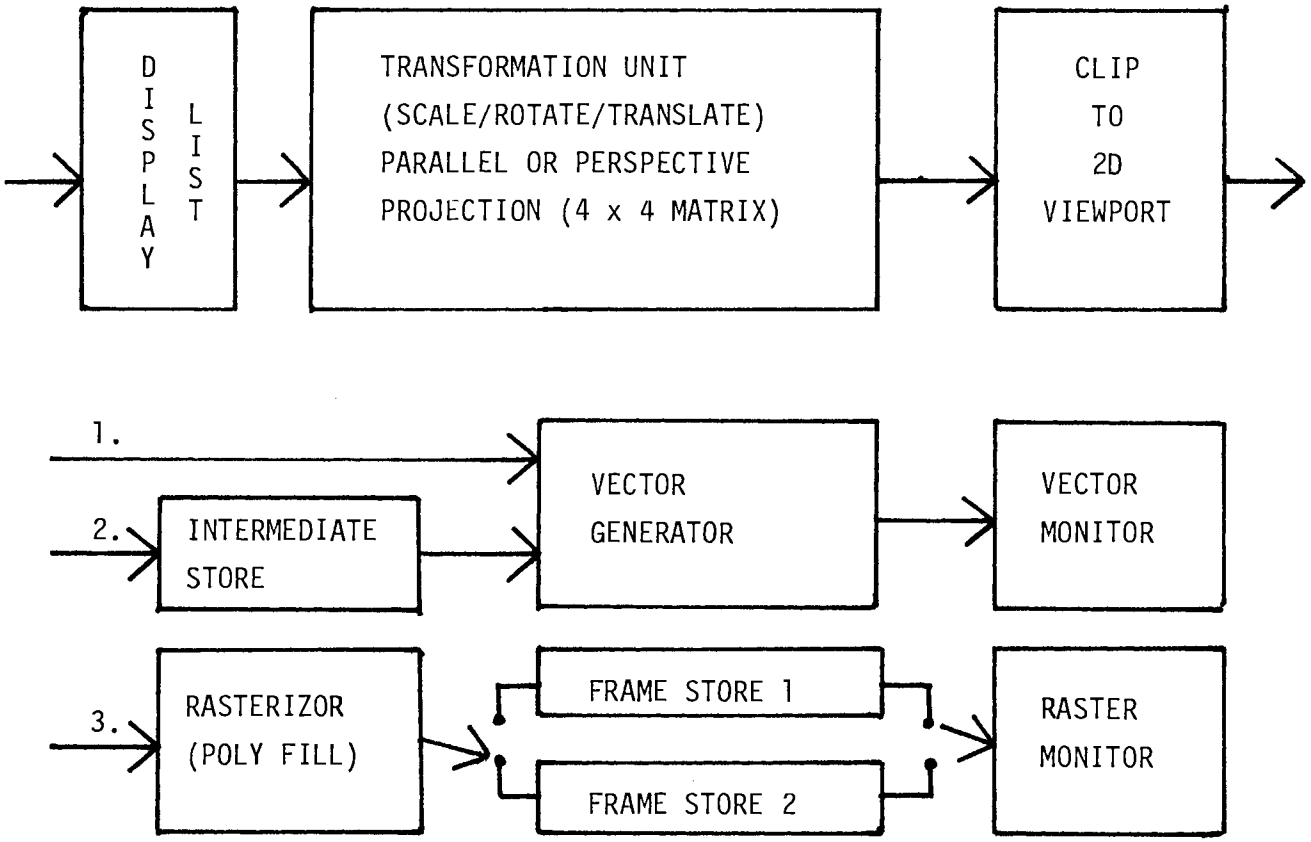


Fig. 4 - 3D Dynamic Display Architecture

The refresh rate must be sufficient such that there is no visible flicker with the speed of phosphor used in the CRT. The minimum considered acceptable is 30 Hz. Note that the eye does not have a linear response to refresh rate. Thus changing from 50 Hz to 60 Hz provides an improvement closer to 50% than 20%. Furthermore, the response to flicker is very sensitive to screen luminosity; the perceived flicker may be reduced by turning down the brightness [6]. As usual there is a trade-off to be made. If the phosphor speed is slow then the picture will be stable, but any movement will be badly smeared.

Vector refresh systems are normally monochrome with up to 16 brightness levels (it is difficult to distinguish more than about 6). In principle, they may be used in conjunction with phosphor penetration CRTs giving 3 colours, but these are not very common. Apart from cost, penetration CRTs have several disadvantages, not least of these being that the writing speed is slower than a monochrome CRT and the electron spot size is larger.

The resolution of monochrome vector displays is excellent, 4096 * 4096 being quite common. There are also models available with sophisticated high-speed processors, allowing real-time manipulation of up to 10,000 or more 3D vectors. Prices start at around SF 30,000 for simple 2D systems, to SF 80,000 and up for systems with 3D capability, depending on the options and input devices required.

2.2.3 Refresh Raster Displays -

Raster displays are not dissimilar in operation to television except that, in general, the scan rates and resolution are higher. This means that the monitors are not driven by standard composite video signals, but by separate red, green, and blue channels. [Very low resolution systems do exist using television monitors]. The simplest systems use a frame memory with a single bit-plane and often have a compatibility mode in which they accept standard Tektonix commands. There is now a large variety of such terminals on the market, with a bewildering number of extra features to support selective erase, colour, area fill (i.e. shading of polygons), etc. An important advantage is that the screen brightness is sufficient to allow use in normal lighting conditions.

There are both colour and monochrome raster terminals available with about 400 * 600 up to 1000 * 1200 addressable points. There are even reports of a monochrome display with 1600 * 2000 points coming soon. Starting prices for Tektronix emulators are from about SF 8,000 for monochrome and SF 11,000 for colour. Software portability problems with displays of this type clearly exist when use is made of their non-Tektronix features.

Moving up the price scale, colour raster displays exist with display lists and quite powerful graphics processors; both bit-slice and standard chips, such as the M68000. This is certainly the area to watch for future improvements in price and functionality. The advantages of a display list are faster response, plus the fact that memory and cpu cycles are saved on the host computer.

At the most expensive end (not counting 'Rolls Royce' systems from Evans and Sutherland) colour raster displays are now available to replace the monochrome CRTs of dynamic 3D vector graphics systems. These never flicker due to the time taken to process the display list, as the screen is updated from the intermediate frame store. To produce smooth dynamics, systems such as that from Megatek use double buffered frame memories. One is filled from the display list whilst the other is used to refresh the screen. The next time through the display list the frame memories are swapped. Thus, at worst, if the display list is very long then dynamic motion may appear jerky. The price of such systems currently approaches SF 200,000 but is coming down.

Because raster display systems will eventually take over almost the whole market, apart from a few specialized sectors, some technical points will be covered in more detail.

2.2.3.1 Considerations On Flicker - It has already been pointed out that for a 1024 * 1024 display to be refreshed at 60 Hz requires a bandwidth of 65 Mbits/s. This means fast memory chips and/or a wide bus, plus a very fast CRT. Manufacturers have tried to reduce the cost of low-end systems in various ways: for example, by lowering the refresh rate and using long persistence phosphors, or by using interlaced line scanning rather than repeat-field. The last approach halves the required bandwidth, but at a significant degradation in picture quality. The trouble is that although television pictures do not suffer overmuch from scan-line interlacing, the effect on graphics displays is much worse. This is mainly because computer-generated graphics often contain single pixel wide horizontal lines, which is almost

never the case for television. As such lines are only refreshed every second picture scan, the flicker may well be unacceptable for prolonged use. Moreover, as graphics images are often stationary for long periods, the effect is not masked by constant motion as it is on a television picture.

2.2.3.2 Colour Convergence - This problem also appears worse than on a normal television. It arises because each pixel is displayed by firing three cathode-ray guns at an adjacent set of spots or lines of differently coloured phosphor. Clearly, it is difficult to ensure that all three electron beams 'converge' to the correct region, especially in the corners of the screen.

The effect one sees on cheap or poorly adjusted colour display monitors is that whilst the picture is reasonable if drawn in primary colours, compound colours, especially white, are fuzzy or fringed. This is particularly annoying if the display monitor is used as a computer terminal for file editing, as characters are difficult to read. The cure is three aspirins, or a more expensive monitor!

2.2.3.3 The Staircase Effect - A serious problem with raster displays is that lines at shallow angles to the verical or horizontal exhibit the appearance of steps (not even regular ones). This effect never occurs on vector displays, even if the number of addressable points is low. It is less important for displaying text, or simple histograms, but can be very disturbing when plotting tracks from an event. The effect clearly improves if there are more addressable points on the display, but even 1000 * 1000 is on the limit for scanning high-multiplicity events.

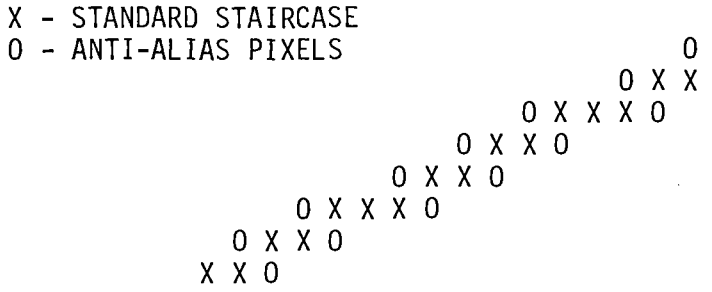


Fig. 5 - Staircase Efect and Anti-Aliasing

A new piece of jargon is the term 'anti-aliasing'. This is a hardware modification (it would be much too slow in software) added by some manufacturers to smear out vectors by filling adjacent pixels at a lower intensity. It does, indeed, make the vectors look more straight, but at the cost of making them 2 to 3 times as wide. Hence this may be useful for computer-generated art, or even drawing pie charts of the CERN budget, but is not really a replacement for a high-resolution display.

2.2.3.4 Image Transforms - Raster systems with display lists may well allow true transfomations of the picture without re-drawing from the host computer. In this case the vectors making up the picture are re-calculated and stored in the frame memory. However, many cheaper systems offer a variety of image transformation functions by 'sort of cheating'. The simplest feature to implement is ZOOM. To do this each pixel in the frame memory is repeated 2, 3, or more times. Thus the picture is magnified, although no extra detail becomes visible.

The possibility to ZOOM is usually accompanied by ROAM, which allows the user to choose the position of the picture. For example, if the display contains 512 * 512 points, and the zoom factor is 4, then it is possible to make any Window of 128 * 128 pixels fill the screen. Moving the window may or may not cause the image to 'wrap around', depending on the implementation. Some systems actually provide frame memories which can store more pixels than can be displayed simultaneously. For example, with 4096 * 4096 pixels and a 512 * 512 display one can ROAM over 64 times the visible area.

Another very popular (and useful) feature of raster displays is the 'polygon-fill' command. This normally requires a point to be given within any closed polygon. The display processor then fills all pixels within the polygon with a specified colour. It is sometimes possible to outline the polygon in a different colour from the interior.

2.3 INPUT DEVICES

Here we list the input devices commonly in use with interactive display systems.

i) Joystick
This is a vertical rod with two degrees of freedom and is used to position a cursor on the screen. Moving the

joystick may move the cursor directly or, in some cases, change the rate of movement. In the first case the precision is usually insufficient; in the second case one normally overshoots the point being aimed at. Joysticks may have a button with which the program can be interrupted.

ii) Keyboard
Keyboards make inelegant graphics input devices. Nevertheless, they often have keys which may be used to move a cursor on the screen. Function keyboards are common input devices for situations where the functionality is static, as single keystrokes are used to specify complete actions. The functions to be performed may be engraved on the keys, and those keys currently enabled can be illuminated on some devices.

iii) Light-Pen
The light-pen is now out of fashion as it not very precise and is tiring to use for long periods. It consists of a photocell which, when pointed at the screen, gives a pulse as the CRT beam crosses its field of view. Thus the device may be used to indicate at which object the user is pointing.

iv) Mouse
If the light-pen is 'out', then the mouse is 'in'. The device is called a mouse because it consists of an elongated hemisphere which is connected to the display via its 'tail'. The top surface may have between one to three buttons, and underneath the mouse can roll on three ball bearings. The rotation of one of these bearings is digitized about two axes, and the output used to drive a cursor. The device is both precise and comfortable to use, as the 'origin' can be moved to any convenient position simply by lifting the device off the desk and moving it. [For this reason it is unsuitable as a digitizer].

v) Tablet
This device consists of a square board on which can be moved a special pen or puck with cross-hairs. Using a magnetic pick-up, or via other means, the position of the pen or the cross-hairs may be accurately determined. Thus the device may be used as a digitizer, or simply to move the screen cursor. Models are available in many sizes and resolutions. It might seem that the tablet would be of more use than a mouse, and this is certainly the case if one wishes to digitize a drawing. However, for simple cursor positioning the tablet is much too

cumbersome (and expensive).

vi) Touch Screen
A Touch Screen normally consists of a thin transparent panel fixed over the face of a CRT screen. Using a capacitance effect, the panel is divided into an array of regions which are sensitive to being touched by a finger. The CRT is used to draw boxes containing menu items which may then be chosen by touching them. Thus the effect is like a set of function keys whose labels may be changed dynamically.

vii) Thumb Wheels
These are two wheels set a right angles into the keyboard of some types of display terminals. They vie for the title of Least User Friendly Way To Move A Cursor!

viii) Valuator
This is typically a box with a set of knobs, the positions of which may be read by the computer. The values they provide may be used to control intensity, position, angle, or whatever.

2.4 BIT-MAPPED WORKSTATION DISPLAYS

This topic has been included as a separate section because Personal Workstations with bit-mapped graphics displays are going to become increasingly common during the next few years. The market is being approached from two sides: from above one is starting to see bit-mapped display systems being added to 'super minis', such as the Digital Equipment VAX; from below, increasingly powerful processors with bit-mapped displays are being offered by home computer vendors, in particular the 'Lisa' machine made by Apple. An interesting point is that the new Apple has by far the nicest user-interface of any current product, aside from SMALLTALK which is somewhat specialized. This should have the very beneficial effect of pushing the more established companies into throwing away their existing maze of control characters.

The bit-mapped workstation display architecture relies on having a frame memory, perhaps with colour look-up tables, as already described. However, in addition, it requires the ability to carry out what Newman and Sproull [7] call Raster Operations (RasterOps), or what Ingles [3] calls BitBlt instructions. The BitBlt got its name from a mnemonic to

specify an instruction which carried out the block transfer of bits. Herein lies the power of the technique.

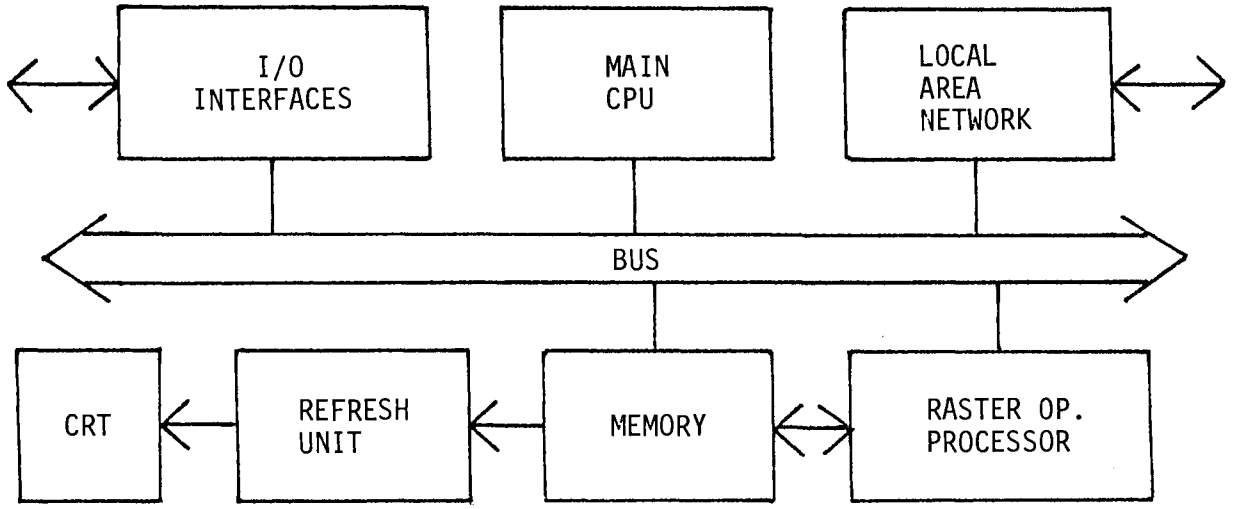


Fig. 6 - A typical Bit-Mapped Workstation

BitBlt instructions allow operations to be performed on rectangular arrays of pixels. Such an operation may simply copy a pixel array from one place to another, or carry out complicated logical operations between source and destination arrays. Apart from being fast (30 Mbits/s/bit plane), BitBlt instructions must have access to a wider address space than only that memory which serves for frame storage. If possible, both the frame memory and data memory should be mapped into the same virtual address space. This allows images to be assembled or stored anywhere in memory before being made visible by moving them into the frame storage area. [The physical address of the frame storage area may be fixed or variable depending on the implementation].

BitBlt instructions provide the possibility of presenting the computer user with a novel way of interacting with the machine. This will be covered more fully in the next chapter but, as an example, one can see that an almost unlimited set of character fonts can be made available. Each character is stored somewhere in memory as a pixel array. When it is required, a BitBlt instruction is used to copy it to the correct place on the screen. Descenders, superscripts, subscripts, and proportional spacing present no problem. Similarly, there is no difficulty in using the BitBlt to scroll text, or anything else, up, down, left, or right.

2.5 HARD-COPY DEVICES

So far we have considered only interactive display systems. However, to work effectively, it is essential to be able to produce 'hard' copies of pictures on paper or film. A major problem in this area is to ensure that the hard copy really is an exact copy of the image on the screen. This may well not be the case if, for example, hardware-generated characters are used on devices which have dissimilar fonts. Below are listed the types of hard-copy devices available, along with their principal characteristics.

2.5.1 Cameras -

Although it is possible to use a Polaroid camera to photograph a display screen, the results are not likely to be of high quality, if only because the screen is not flat. However, devices are available with a high-precision flat-faced CRT which is driven from the same red-green-blue (RGB) signals as a refresh raster display system. They use a black and white CRT, which can always produce better pictures than a colour one, and make three exposures through red, green, and blue colour filters. Most devices accept a range of cameras and film formats from 35 mm to 9 * 13 cm and A4, using negative, reversal, or Polaroid film. (Note that Polaroid now make 35 mm film). An option of particular interest is the ability to put a set of 35 mm size images on one piece of A4 size Polaroid transparency film. This is both a cheap and very fast way to prepare the slides for a presentation.

It is not always easy to match the signals from the display processor to the copier as there are lots of variables to consider, so take care. However, camera systems have the very important property that 'You Get What You See'. Prices start from about SF 30,000 and go up to SF 60,000 +.

2.5.2 Impact Printers -

There is now a range of Dot Matrix impact printers available which are capable of producing graphical output. They are slow and have only a moderate resolution (around 100/130 dots/inch). Models exist which can print up to six colours. Depending on speed and resolution, prices are from SF 8,000 and upwards.

2.5.3 Ink Jet Printers -

Although it may sound improbable that it is possible to mass produce a printer which works by shooting jets of different coloured ink at a piece of paper, this is in fact the case. Like impact printers, they also work on a dot matrix principle and models exist which produce medium- to high-resolution output in up to eight colours. Prices start even below SF 5,000 for very slow, low-resolution models.

2.5.4 Laser Printers -

Alphanumeric laser printers, which already produce characters using a dot matrix, have had controllers adapted to produce high-quality monochrome graphics (up to 200 dots/inch). They operate much faster than impact printers, but prices currently start around SF 50,000.

It is interesting to speculate how long it will be before a colour device appears. One manufacturer has already exhibited a colour copier.

2.5.5 Electrostatic Printers -

Electrostatic printers have been one of the standard hard-copy devices for a long time. They have been steadily improved and models now exist with very good resolution (up to 200 dots/inch). New plotters are coming with colour, but these are both complicated and expensive. Black and white versions are available with paper widths from 11 to 40 inches, and prices start at around SF 20,000.

2.5.6 Pen Plotters -

The original computer graphics output devices, pen plotters, are now available which are the size of a briefcase and take A4 paper. Versions with 6 different pens cost from SF 10,000. The main problem is that they are very slow.

2.6 THE CRYSTAL BALL DISPLAY

This is a special display at which one looks to see what will happen over the next 2 to 5 years! Writing predictions down on paper is asking for trouble but, nevertheless, here goes ...

i) Price breakthroughs may come when the market gets large enough for real mass production. This has happened for the alpha-numeric terminal market already, and may well occur for the mid-range display market over the next few years. It cannot be too long before Japanese products start to make a significant impact.

ii) CAD/CAM applications will expand the top end of the market, with more products offering dynamic 3D with hidden line and surface removal. Systems offering surface shading in real-time will become more common.

iii) As systems offering solid modelling take over at the high end of the market, the functionality offered by products currently in this position will start to become available in the intermediate price category. More mid- to top-range products will appear with multi-level segmentation and floating-point format for coordinate storage.

iv) There is less room for large reductions in low-end prices, but as more powerful processors become available and memory gets cheaper, one will continue to see increased functionality in this range.

v) One area to watch is the appearance of 'quasi' 3D dynamic wire-frame displays. These will be colour raster systems with display list memories and a cpu with 3D software. They may take several seconds to produce a picture with a few thousand 3D vectors, but this will improve with the processor clock rates. This is a very interesting area for HEP applications.

vi) As the number of units over which development costs can be divided increases, and VLSI design becomes easier, display systems will become available with purpose-built chips. The Clark Geometry Engine [8] is the first such device, although it is not yet commercially available. It can process 6000 end-point calculations per second in floating-point. Clark suggests that a system could be configured with a pipeline using 12 devices. In this case the performance would be sufficient for dynamic 3D transformations.

vii) As the market for graphics laser printers increases one will certainly see the costs fall; especially when the manufacturers start to incorporate the graphics

controllers directly in the printers.

viii) There is still some interest in flat screens, and a 5-10 cm thick CRT seems possible. Solid state displays with 500 * 500 pixels should also appear in this time frame.

To sum up then, the computer graphics market will continue its current expansion over the next few years. The overall result for the user will be more functionality and more choice for less money. The problem is that many of these nice new devices will be made by companies which are small, and/or are not based in Europe. Perhaps purchase price will be less important than hardware and software maintenance!

CHAPTER 3

BASIC GRAPHICS SOFTWARE

3.1 GRAPHICS CONCEPTS

Before going more deeply into computer graphics software, we begin with brief explanations of some concepts or definitions which are not covered elsewhere in this paper (see also Ref. [9], pages 9,11).

3.1.1 Aspect Ratio -

This is the ratio of the X and Y extensions of a screen, Window, or Viewport. For the aspect ratio to be preserved in a transformation the X and Y scaling must be the same.

3.1.2 Clipping -

The action of clipping (or scissoring) removes those graphics primitives (or parts thereof) which lie ouside the Window or Viewport boundary.

3.1.3 Coordinate Systems -

There are three different coordinate systems used to describe the positions of Graphics Primitives: World Coordinates, Device Coordinates, and Normalized Device Coordinates (NDC). The use of the first two of these systems is straightforward. The application programmer chooses a World Coordinate scale suitable for describing the object or

model to be drawn. The Device Coordinate system is that used by a particular piece of hardware, for instance from 0 to 511 or from -1023 to +1023.

Some graphics packages, including GKS, also specify Normalized Device Coordinates in the range 0 to 1 as an idealized description of a display surface. The position of a Viewport on the display surface may be specified in NDC and makes the application program independent of a particular hardware device. The contents of a Viewport are mapped from NDC to Device Coordinates via the Workstation Transformation without changing the Aspect Ratio. The mapping of World Coordinates onto a Viewport is carried out by a Normalization Transformation, of which there may be several (see Window).

3.1.4 Current Point -

The concept of the 'current point' dates from the use of pen plotters. It was then meaningful for a graphics package to move the current point (i.e. pen) to a particular position before drawing something. Moreover, after drawing a line on the paper, the current point clearly corresponded to the line's end coordinates. However, it is less obvious where the current point is supposed to be after, for example, a polygon-fill instruction to a refresh raster display.

Thus modern graphics packages tend to no longer include the current point concept: it is always necessary to specify the starting point for a Graphics Primitive. However, this does not mean that one necessarily has to specify the first and last point of every vector. Packages invariably include a 'poly-line' primitive so that a sequence of adjoining vectors may be specified by giving the starting point of the first vector plus all the end points.

3.1.5 Device Driver -

This is the code which sits between a graphics package and the display hardware. It handles the interface to the operating system, interrupts from the device, and such like.

3.1.6 Graphics Attributes -

Graphics attributes describe the way in which Graphics Primitives appear. They include:

i) Style of font.

ii) Character spacing, angle, etc.

iii) Colour or/and intensity.

iv) Line style (e.g. width, dashing).

v) Highlighting (e.g. blinking).

vi) Visibility.

vii) Pickability: some graphics packages allow a 'pick identifier' (pick-id) to be stored as a special primitive attribute (see section on picking, below). A pick-id is a special attribute beacuse it only has meaning to the application program.

viii) Transformation: a modelling transformation applied to a graphics primitive before it is drawn may also be considered to be an attribute.

3.1.7 Graphics Primitives -

Graphics Primitives are the basic building blocks from which pictures are built. They include:

i) Poly-Line: a set of adjoining vectors specified by the starting point of the first vector, plus all the end points.

ii) Poly-Gon: as above but with the first and last points connected. (N.B. Does not exist in GKS.)

iii) Poly-Marker: a set of points at which a marker symbol is drawn. (A marker symbol could be a point, cross, diamond, square, etc.)

iv) Text: a starting point followed by a text string.

v) Fill-Area (or polygon-fill): fill an area with a solid colour or cross-hatching.

vi) Pixel-Array (or cell array): set the bits in the frame memory of a raster display with the given array of pixels. This primitive is a rather inelegant way of retro-fitting some ability to deal with bit-mapped raster displays into vector-oriented graphics packages. The results are not likely to be very portable.

vii) General Drawing Primitive: a standard way to be non-standard! Some graphics packages (including GKS) include this primitive as a way of accessing special hardware functions.

3.1.8 Meta-Files -

Meta-files are data structures in which may be stored the description of a (set of) picture(s). For example, if working on an interactive display, it should be possible to produce a meta-file describing a picture which has been generated. At a later stage, the meta-file may be read back and the picture reconstructed. It should also be possible to use a meta-file to produce copies of the picture on other displays or hard-copy devices. A standard meta-file format is clearly essential for transmitting pictures between different computer systems.

3.1.9 Picking -

Most users of graphics terminals will be familiar with the use of a light-pen or cross-hair cursor to 'pick' an object on the screen. The reason for mentioning it here is that previously the work was done by the application program rather than the graphics package. The cursor returned only a position in screen coordinates and it was up to the package to find out to which object this corresponded.

More modern packages retain a 'locator' device to provide the cursor position (perhaps even in World Coordinates). However, in addition, there exists a 'pick' device which tells the application program which object on the screen was picked. This is only possible if the package has a data structure describing the picture. Thus the pick device can return, for example, the name of the segment containing the picked object along with the particular primitive within the segment at which the cursor was pointing. Some packages also allow the application program to store private pick information, called a 'pick-id', in segments (primitives) which is returned if the

segment (primitive) is picked. Thus the application program has several ways to relate the picked segment to its own data structure.

3.1.10 Segmentation -

This will be covered in some detail, as the concept may be novel to those people unfamiliar with refresh displays. When using storage-tube displays or hard-copy devices it is not necessary to have a graphics data structure because it is not possible to interact with the picture. If anything is to be changed then the whole picture must be re-drawn. However, with refresh systems this is not the case. It is perfectly possible to erase a particular vector, or to change the colour of a single line of text. The problem is how to address those primitives which are to be modified (deleted, picked, ...).

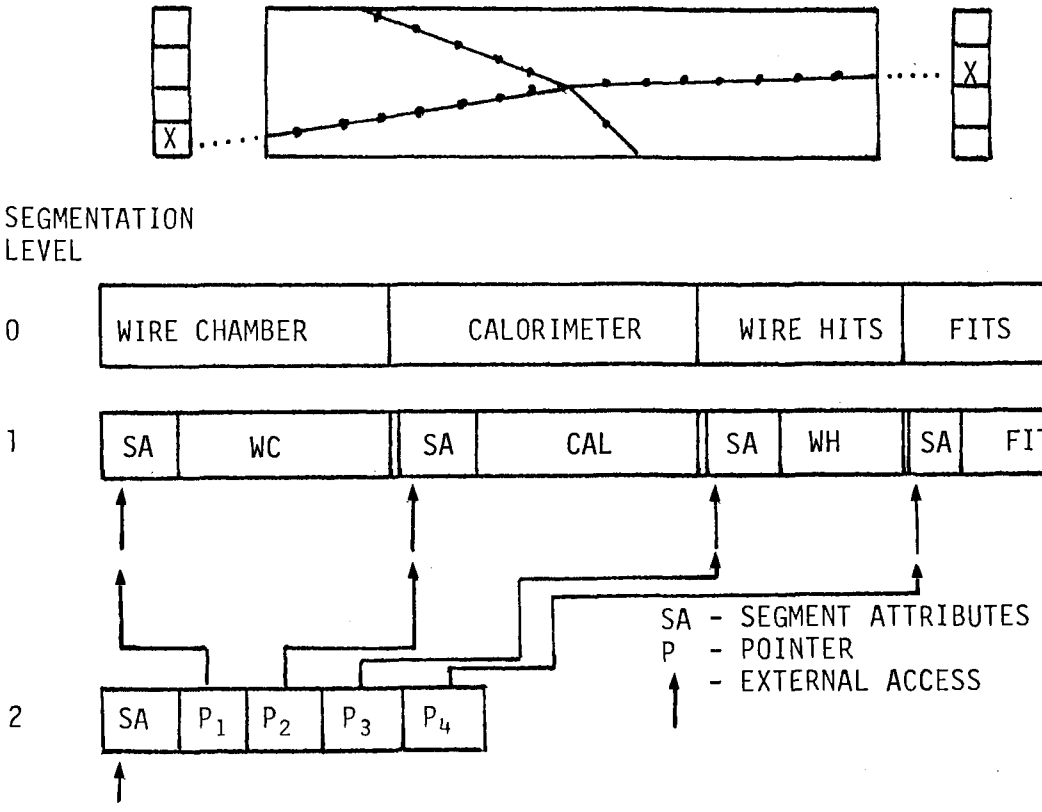


Fig. 7 - Simple Graphics Structures

The diagram shows three cases. For case (0) the picture is stored without any segmentation. Any change requires all the primitives to be re-drawn. However, case (1) implements a single level of segmentation, like GKS for example. The

primitives may be stored by the hardware but this is not necessary. For example, even using a DVST it is useful for the graphics package to store primitives in segments. This is because, even if the screen has to be erased and re-drawn to modify the picture, the application program only has to re-create those segments which are to be altered.

As described elsewhere [1,10], graphics data structures may be quite complicated. Only one further point will be described here which relates to case (2) of the diagram. This shows a second level of segmentation, and the point to be made is that this is the minimum level of segmentation necessary for dynamic transformations. The reason is that if the image represents a solid object, any transformation must be applied simultaneously to all its constituent segments. If only a single level of segmentation were employed, then it would be necessary to update the transformation matrices in each segment header separately. This is at best inconvenient and clearly impossible if the goal is the smooth dynamic movement of several hundred segments. The problem goes away if a single matrix is stored at the next higher level and may be applied to all segments lower down.

3.1.11 Transformation Matrix -

By multiplying its coordinates by a matrix it is possible to transform a point by translation, scaling, or rotation about an axis. Using homogeneous coordinates and a 4 * 4 matrix it is also possible to perform parallel or perspective projection of a 3D point onto a plane. It is often advantageous to combine the rotation, scaling, translation, and projection matrices into a single 'transformation matrix' before transforming a set of points.

3.1.12 Viewport -

A Viewport is a rectangle on the display surface which is usually specified in Normalized Device Coordinates.

3.1.13 Virtual Camera -

The concept of the Virtual Camera was used by the SIGGRAPH Core System to map the contents of a 3D window onto a display surface. The idea is to imagine the Viewport as a

camera pointing at a three-dimensional object situated in a Window. The specification of all the relative viewing positions and angles will not be elaborated here.

3.1.14 Window -

A Window is a rectangle specified in World Coordinates and is mapped by the graphics package onto a Viewport. If the Viewport is specified in NDC, then the Window to Viewport transformation is called a Normalization Transformation. There are as many Normalization Transformations as there are Window-Viewport pairs. Note that the concepts of Viewport and Window may be extended to three dimensions, but with a good deal of complication in the mapping process (see Virtual Camera).

3.2 WHAT SHOULD A GRAPHICS PACKAGE PROVIDE?

It must be stated straight away that a Graphics Package is a tool used by the graphics application programmer. The user of a graphics system often requires access to a higher level of functionality to solve particular problems. The application programmer may also require other tools for such things as histogramming, menu selection, contour plotting, etc., which are supplied by libraries situated on top of the basic graphics package [11-13].

Thus, the basic graphics package should provide:

i) The possibility to drive different types of graphics display without changing the application program. This includes a standard way of specifying graphics primitives (Poly-line, Text, ...) and attributes (fonts, line-styles, ...).

ii) The ability to map World Coordinates (floating-point) onto the display hardware. This should include multiple Viewports, Transformations, and 3D to 2D projection.

iii) A standard way to handle many types of input devices, along with the ability to enable different device sub-sets simultaneously.

iv) The storage of picture segments so that it is possible to make changes without re-drawing everything. Multi-level segmentation is required to support dynamic displays.

v) A meta-file for long term storage and/or transmission of images.

3.3 GRAPHICS STANDARDS

The first efforts in the direction of Graphics Standards were made in 1974 when ACM SIGGRAPH set up the Graphics Standards Planning Committee (GSPC). This committee first published results in 1977 with a prototype 'Core System' which was supposed to encompass all the basic requirements for a 3D graphics package. At this time the GSPC also set up sub-committees to look into raster and colour graphics, meta-files and extensions to the Core System. A final report appeared in 1979, during which year the X3H3 committee was set up by ANSI to turn the GSPC Core System into a proposal for an ANSI standard. This all occurred during a period of great activity in the graphics display market so that, even though not an ANSI standard, the Core System was adopted by many manufacturers.

Meanwhile in Europe a German study group (NI-5.9) set up by DIN had been developing a Graphics Kernel System (GKS) [9]. This was first presented to the International Standards Organization in 1978. During the following four years, in collaboration with ANSI and the British Standards Institute, a great deal of discussion took place on GKS. A substantial number of modifications were made to the original document and finally, during 1982, the technical content of GKS became accepted. The final vote to accept GKS as an ISO standard should take place during 1984.

The most important difference between GSPC Core and GKS is that the latter is only 2D. Work is currently proceeding to produce 3D extensions to GKS, but this is likely to be a slow process. Meanwhile various packages will probably be produced to carry out a 3D to 2D projection on top of GKS. The problem with this approach is that the GKS segment storage cannot contain the original 3D primitives. Thus 3D transformations will not be possible without the application program re-creating the complete image.

Nevertheless, even with the drawback that GKS is 2D and only has a single level of segmentation, it will be the first international standard. As such, it seems to have become accepted by the graphics community (even in the United States) and, over the next year, one can expect to see various implementations appearing from universities and software houses. Hence GKS will become the CERN standard package for

2D graphics.

Whilst on the subject of standards, the Virtual Device Interface (VDI) must be mentioned. This is a proposal by the X3H3 committee (receiving support from the Digital Equipment Corporation, Tektronix, and Intel) to define a standard set of functions to which a graphics device (or its driver) will respond. The wide acceptance of such a standard would clearly help the problem of graphics package portability.

3.4 AN OVERVIEW OF GKS

The ISO specification of GKS [9] covers the functionality which must be supplied by an implementation. Different 'language bindings', which specify the procedure calls in detail, are required for each computer language. The binding to FORTRAN 77 will be a separate ISO standard which is currently being finalized. We now look at the major features of GKS.

3.4.1 Drawing Facilities -

GKS has no concept of 'current point'. The output primitives and attributes are similar to those listed above in the section on Graphics Concepts (not including polygon). GKS also has the concept of 'attribute bundle', which is analogous to a pen. The idea is that a number of attributes can be bundled together instead of being specified separately. An application program which is transferred to a different display could then map facilities from the old to the new display by re-defining the attribute bundles. For example, when moving from a monochrome to a colour display, bundles specifying line style (e.g. dash pattern) could be re-defined in terms of colour. Thus attribute bundles provide dynamic binding of attributes to primitives. It is also possible to assign static attributes to individual primitives, but then to change the appearance of these primitives they must be re-created by the application program.

3.4.2 GKS Workstations -

GKS is based on the the concept of logical workstations which are situated above the device drivers. A workstation can have a certain maximum set of capabilities. These include:

i) a single rectangular display surface,

ii) support for several line types, text fonts, character sizes, etc.,

iii) one or more logical input devices for each input class,

iv) Workstation Dependent Segment Storage (WDSS),

v) three possible input modes: Request, Sample, and Event.

Workstations do not have to incorporate all these capabilities, but they do have to support 'enquiry functions', so that the application program may find out at run time what facilities are available. This enables the program to adapt itself to the current environment in a dynamic way.

The importance of the workstation concept is that the higher levels of the graphics package are presented with a standard interface to all hardware display systems. Workstation code must be provided for each supported display type to match this interface to the facilities provided in hardware. Thus, if the implementation is well done, the workstation should only provide in software those features not in the display. For example, on a raster display, area filling is normally available using a hardware instruction. However, on a vector display, area filling might be simulated in the workstation code by using vectors for cross-hatching.

Workstations may be dormant or activated. All workstations which are currently activated receive the graphics output information generated by the application program. Thus the same information may be made visible simultaneously on several display systems. GKS acknowledges the existence of two special workstations. The first corresponds to the meta-file output. In other words, if this workstation is currently activated, then as the graphics output is generated it will be written onto the meta-file. The second special workstation corresponds to Workstation Independent Segment Storage (WISS), which will be described below.

3.4.3 Segmentation -

At the simplest level, graphics output may be sent straight through GKS directly to a workstation's display surface. In this case it is not possible to make alterations to an image without first deleting it and then re-sending everything. However, GKS provides facilities to group

graphics primitives together within segments, which have unique, application-specified names. The contents of segments may be:

i) transformed (2D rotation, scaling, translation)

ii) made visible/invisible

iii) highlighted (i.e. made to blink)

iv) made pickable or not

v) deleted or renamed

vi) priority ordered (this affects overlapping primitives).

To use a segment it must first be opened, then filled, and finally closed. No facilities are provided to extend segments or alter their contents after they have been closed, except as listed above.

Segments stored by normal workstations are considered to be workstation dependent and are available only to that workstation. This is called Workstation Dependent Segment Storage (WDSS). However, segments stored in Workstation Independent Segment Storage by the special WISS workstation may be manipulated as follows:

i) They may be COPIED to another workstation (which presumably was not activated when the segment was originally created). In this case the segment is displayed by the workstation but not stored in its WDSS.

ii) They may be ASSOCIATED with another workstation. This has exactly the same effect as if the workstation had been activated when the segment was originally created.

iii) They may be INSERTED into the currently open segment. This has an effect similar to that which would occur if the contents of the inserted segment had come directly from the application program.

3.4.4 Logical Input Devices -

The concept of a Logical Input Device is intended to provide the application program with a standard interface to each class of device. As an example, the logical LOCATOR returns the World Coordinates of the screen cursor in a well-defined format. However, the physical representation of

this device could be a tablet, joystick, mouse, etc. Thus the application program does not have to incorporate directly code to handle particular devices and so becomes more easily portable. [There are one or two dissenting voices in the graphics community which point out that this works both ways: there is no standard way of making use of any special features provided by a physical device.]

As far as we need be concerned here, GKS defines each class of logical device in terms of the following:

i) Measure: This is the information provided by the device (i.e. coordinates, character string, etc.).

ii) Trigger: This specifies when the measure should be acquired. Thus the trigger for a character string might be the typing of 'carriage return'.

iii) Prompt/Echo Type: This describes how the operator should be prompted for input and what form the echo should take.

Logical input devices are clearly implemented at the level of the workstation, as they depend completely on the facilities provided by the hardware. A workstation classified as either an Input Workstation or an Input/Output Workstation must provide at least one logical device of each class. The classes are:

i) LOCATOR: provides the position of the cursor in World Coordinates,

ii) STROKE: provides a sequence of cursor positions (to be used for curve drawing, for example),

iii) VALUATOR: provides a real number,

iv) CHOICE: provides an integer selection from a range of choices (i.e a menu or function keyboard),

v) PICK: provides the name of the picked segment (plus more details),

vi) STRING: provides a character string.

GKS provides rather sophisticated support for obtaining input for the application program. Logical Input devices may operate in three different modes:

i) REQUEST: This mode waits for input to be obtained from a particular logical device.

ii) SAMPLE: This mode reads the current 'measure' of the specified device, without waiting for a 'trigger'.

iii) EVENT: Input from logical devices operating in EVENT mode is stored in an Event Queue maintained by GKS. There are functions provided to interrogate the queue and also to clear it, delete an entry, etc.

3.4.5 GKS Meta-file -

As already described, GKS provides a mechanism to store graphical output on a sequential meta-file. The format of the file is not defined by the ISO standard, but forms an annex to the main document. Apart from graphical information, the meta-file may store private information generated by the application program. The application program must take care of such information when the meta-file is re-read as, clearly, GKS has no way to interpret it.

Once a meta-file has been closed, it may be re-opened for reading. The functions provided for this are:

i) GET-ITEM-TYPE: Read the next meta-file record and decide whether or not it may be interpreted by GKS.

ii) READ-ITEM: Read the next meta-file record.

iii) INTERPRET-ITEM: Causes the current meta-file record to be processed by GKS as if the information was being supplied directly from the application program.

3.4.6 GKS Implementation Levels -

In order that low-level applications may make use of GKS without incurring a penalty and that GKS can be implemented on small computers, the GKS standard has been defined at different levels. Thus one would expect lower levels to be less costly to purchase and to consume less computer resources than the higher ones. The levels have been divided into a matrix with increasing output functionality along one axis and increasing input functionality along the other. The matrix of levels has been summarized below.

	A	B	C
0	NO INPUT ALL OUTPUT EXCEPT FOR SOME ATTRIBUTES	REQUEST INPUT NO PICK	SAMPLE + EVENT NO PICK
1	COMPLETE OUTPUT BASIC SEGMENTS META FILE MULTIPLE WORK- STATIONS	REQUEST WITH PICK	SAMPLE + EVENT WITH PICK
2	WORKSTATION INDEPENDENT SEGMENT STORAGE (WISS)	-	-

Fig. 8 - A Summary of GKS Implementation Levels

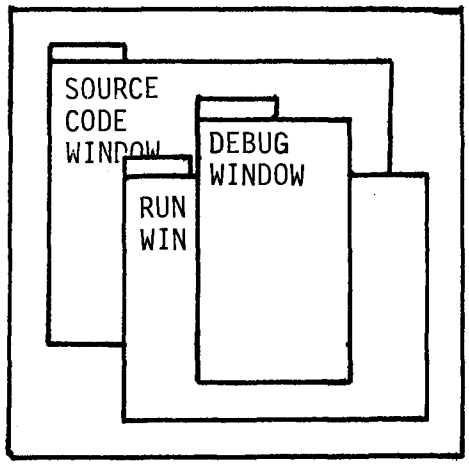
3.5 SOFTWARE FOR BIT-MAPPED GRAPHICS WORKSTATIONS

Workstations with bit-mapped graphics (and RasterOps) are only now starting to become generally available. With them comes the possibility of providing the user with a whole new way of using a computer. In this section we will look at some of the features which the software can provide. It is assumed that the hardware level includes a screen with at least 1000 * 1000 dots and a mouse as a pointing device.

3.5.1 Screen Management -

In most graphics packages the display surface is assumed to be the property of the application program. The latter may be provided with facilities to support multiple Viewports on the same screen, but if one overlays the other, then the result is likely to be a mess. If two separate processes output data to the same screen, then the result is guaranteed to be a mess (unless they agree beforehand to use only their own areas).

In 1972, the SMALLTALK system, developed by Alan Kay and others [3,4], included the concept of overlapping windows (the nomenclature is confusing, today they would be called viewports). The idea behind this was that one normally works on a desk covered with papers (to an arbitrary depth!). Thus, it was reasoned that it should be possible to work with a computer on several projects at the same time. If the CPU could be time-shared, then why not a terminal? To take an obvious example, during program development the programmer would like simultaneous access to the source code, the compiler output, the linker map, the running program, and also an interactive debugger. SMALLTALK was the first environment in which this all became possible.



- WINDOWS MAY BE MOVED
- THEY MAY CHANGE SIZE
- ANY WINDOW MAY BE PLACED "ON TOP" OF ANY OTHER
- WINDOW CONTENTS MAY BE SCROLLED IN X AND Y

Fig. 9 - Overlapping Windows

These ideas are now being incorporated into Personal Workstations. The usual implementation is for each service required by the user to run in its own private process. Thus the example given above would require the programmer to have five processes running; equivalent to logging in five times on a time-sharing system.

Each process requires interaction with the programmer, presumably not via five terminals! The answer is that each process has access to a virtual display screen. These virtual screens are coordinated by a piece of software called a Window Manager. It enables all the virtual screens to be visible simultaneously, perhaps with some partially overlaying others. The user can only interact with one process at a time, but it is possible to choose that process at will by pointing to it. The Window Manager normally ensures that the process enabled for interaction has its virtual screen 'on top' of the others.

The Window Manager provides the user with several functions apart from being able to choose the window enabled for interaction. Facilities are available to change the size and position of any window, or scroll the contents of windows up or down and left or right. This functionality is made available by maintaining bit-maps or display lists in memory corresponding to each virtual screen. The Window Manager operates by using these bit-maps to build up the image which actually gets mapped to the screen.

3.5.2 Mixing Text And Graphics -

It has already been described, in the chapter on hardware, how bit-mapped displays may support an almost unlimited variety of different fonts. These can be used by very sophisticated full-screen editors, so that the user is continuously presented with an up-to-date image of all text and mathematical symbols. Such display systems also support graphics editors which allow the user to 'draw' pictures on the screen and then modify them.

- MULTIPLE TEXT FONTS
- SPECIAL SYMBOLS
- TEXT AND GRAPHICS EDITORS
- "WHAT YOU SEE IS WHAT YOU GET"

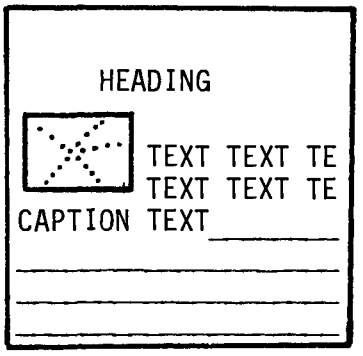


Fig. 10 - Mixed Text and Graphics

The drawing is done by moving a 'pen' with the mouse. A selection of pens may be available which have different thicknesses or colours. Features may also be provided to help draw schematic diagrams. For instance, a library of pre-existing symbols may be available, or a rectangular net of points superimposed on the screen. By constraining all lines to start and end on one of these points it is possible to make very neat drawings.

The final touch is that the results of the text and graphics editors may be combined and so it is possible to produce a complete document. A hard-copy device, such as a

laser printer, is clearly required to make an exact copy of the screen. If one of the devices incorporating a camera is available, then transparencies may be produced for use at presentations.

3.5.3 A New User Interface -

This is an extensive subject in itself and cannot be covered in any detail. Just two features will be mentioned, aside from the use of the Window Manager described already. It is implicit that a 'mouse' is available to drive the cursor.

3.5.3.1 The Use Of Ikons - In the jargon, an Ikon is a symbol (a small picture) used to convey information. It may seem like a game, but the use of Ikons saves space and time. They are also much easier to understand for inexperienced users than the normal prompt messages. Here are some examples from SMALLTALK or the Xerox Star:

i) The whole filing system may be hidden behind a set of Ikons representing filing cabinets, box files, folders, etc.

ii) A waste-paper basket to represent the delete action.

iii) An hour glass showing what fraction of the total time needed to perform an action has passed.

iv) An envelope representing the mail system.

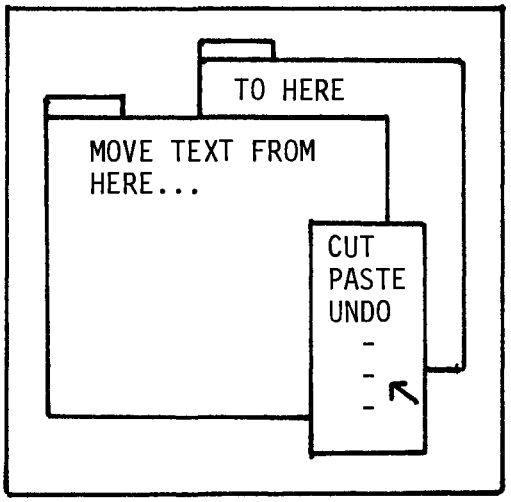
v) The cursor turns into an arrow indicating the position at which text will appear if the keyboard is used.

vi) The cursor turns into an ellipsis (...) indicating that the user must wait for the computer.

vii) The cursor turns into a pair of spectacles indicating that the computer is reading a file.

3.5.3.2 Pop-Up Menus - These make use of the buttons on the mouse. First one moves the cursor to a free piece of screen and presses a button. A menu of items then appears on top of what was there before. If the cursor is moved out of the menu area and the button released then all that happens is that the menu goes away. However, whilst within the menu area the item

pointed at by the cursor is highlighted in some way. If the button is released at this point then the item is chosen and the menu disappears, leaving the screen in its original state. SMALLTALK always has an UNDO option in all menus to cancel the effect of the last action.



- MENU APPEARS ON PUSHING A BUTTON ON THE MOUSE
- OPTION POINTED TO BY CURSOR IS HIGHLIGHTED
- WHEN BUTTON RELEASED OPTION IS CHOSEN AND MENU DISAPPEARS

Fig. 11 - Pop-Up Menus

As an example of the power of this feature, it is possible to CUT a piece of text out of one file and PASTE it into another with just six button pushes and no typing.

3.6 PIONS

This review of software will be completed by a short description of the PIONS graphics package. The acronym stands for Partial Implementation of Our New System, which provides an introduction to its origin.

In the spring of 1980, a VAX 11/780 computer and two Megatek monochrome 3D vector display systems were purchased for use by the UA1 and UA2 experiments. The installation was called Merlin. No suitable 3D graphics package existed and so a subroutine library, called PIT, was implemented as a medium-term solution. The motivation for developing PIONS came from two sources. The first was the realization that no standard 3D package which met our needs was available, or was even foreseeable within a reasonable time scale; the second was the experience gained from the use UA1/UA2 were making of the Merlin system.

The two main design goals for PIONS were:

i) Full advantage should be taken of display systems incorporating 3D dynamic hardware.

ii) The package should allow a user to view interactively different parts of a detector or an event without requiring the intervention of the application program.

Secondary goals were that the package should support multiple device types, simultaneous use of multiple viewports, simultaneous use of multiple input devices, etc.

PIONS currently runs on a VAX 11/780 and supports both Megatek and Tektronix display systems. Code to make use of the selective erase features of one Tektronix compatible raster display has been added and it is quite possible to further extend the range of supported displays in the future. At present the programs using the PIT package are being converted to use PIONS.

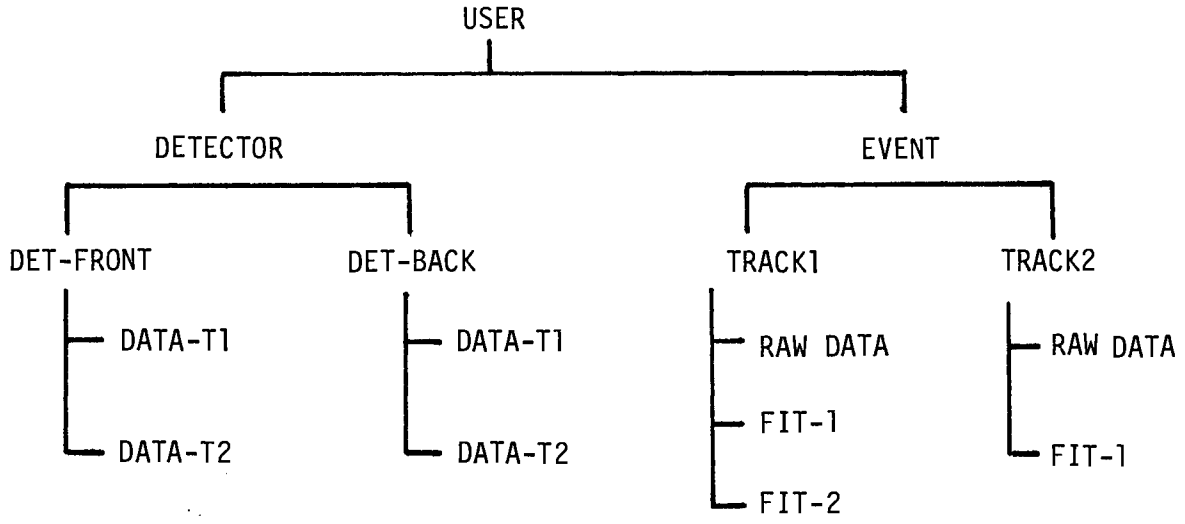


Fig. 12 - A Simple PIONS Data Structure

Perhaps the best way to explain what PIONS does is to describe how it is used to view an event. A basic idea is that PIONS separates the graphical description of an image from the way it is viewed. Thus, to start off with, the application program builds a graphics data structure containing descriptions of the detector, the raw data from an event, and the results of any fits.

In parallel, another structure exists describing the display hardware available and the different viewports the application program has created. There may be many viewports available on several screens and they describe the projection of the 3D model in terms of viewing angles, view reference points, and so on. For a user to view an event, one or more viewports are instructed to display some part of the graphics data structure.

The graphical information is all stored in floating-point World Coordinates. Thus the user of the system may request to view the event with different scales or viewpoints without the application program having to re-supply any information.

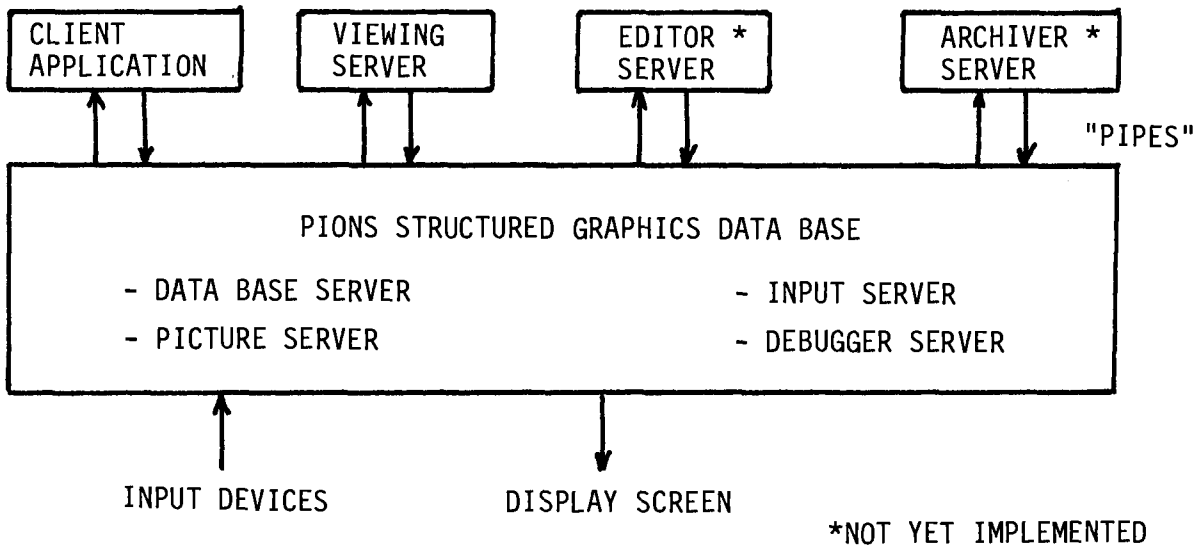


Fig. 13 - Schematic Diagram of PIONS

PIONS is designed to operate as a process independent from the application program. This provides both sides with a certain insulation against changes and also the possibility to run on different processors. The logical structure of PIONS is also based on the client-server principle. Thus the application program is a 'client' of the PIONS process, which provides a range of services by incorporating a set of logical servers. These consist of:

i) a Data Base Server (handling space allocation),

ii) a Picture Server (with a set of data base interpreters, one for each workstation type),

iii) an Input Server (mapping logical to physical input devices),

iv) a Debugger Server,

v) a Viewing Server (allowing interactive viewing of the data base without intervention of the application program).

In the future, it is hoped to add servers for archiving (a meta-file) and picture editing.

It has been mentioned that PIONS could run in a separate processor from the application program. The motivation for this is to try and resolve the mutually incompatible requirements of a package such as PIONS. These are:

i) A fast response time for displaying different aspects of complicated events.

ii) Compatibility between high-performance 3D display systems and cheap graphics terminals.

iii) Operation on a range of host computers.

By moving the graphics package out of the host, it is only necessary to maintain it on one machine. Moreover, the performance should be improved by overlapping the work done by the two processors, and the load on the host is significantly reduced.

CHAPTER 4

CONCLUSIONS

When deciding on a strategy for using graphics the problem is that a trade-off exists between colour, resolution, dynamic 3D, and cost! One can have the lot, but at a price. Thus, it is unlikely that a LEP collaboration, for example, could standardize on a single display system. It will be necessary to consider carefully the proposed applications and then to choose two, or perhaps three, display systems to match. It must also be borne in mind that the market is changing very rapidly. Thus over the lifetime of an experiment the equipment will have to be replaced as it becomes unmaintainable. The fact that the replacement systems may not be totally compatible with the old ones will be somewhat offset by their being cheaper and having more functionality.

REFERENCES

[1] J.D. Foley and A. van Dam
Fundamentals of interactive computer graphics
(Addison-Wesley, Reading, Mass., 1982)

[2] Helga E. Rafelski
The CERN Computer Graphics Guide,
CERN DD/US/57 (1979)

[3] BYTE (August 1981), p. 14.

[4] A. Goldberg and D. Robson
SMALLTALK 80: The interactive programming environment,
to be published by Addison-Wesley, Reading, Mass.

[5] R. Brun, F. Carena, M. Hansroul, J.C. Lassalle and G. Patrick
GEANT
CERN DD/US/86 (1982)

[6] Raster Graphics Handbook
(Conrac Corporation, Covina, CA, USA), p. 2-33

[7] W. Newman and R.F. Sproull
Principles of interactive computer graphics
(McGraw-Hill, New York, 1979)

[8] J.H. Clark
A VLSI geometry processor for graphics
Computer 13 (7), 59-68 (1980)

[9] Graphical Kernel System functional description (GKS 7.2)
Draft International Standard ISO/DIS 7942 (November 1982)
ISO TC97/SC5/WG2 N 163

[10] L.J. Oostrijk, D.R. Myers and J. Bettels
PIONS, A high performance graphics package
Internal CERN DD documentation available from the authors.

[11] R. Brun and H. Watkins
HPLOT
CERN DD/EE/80-2, DD/US/17 (1982)

[12] R. Brun
HTV
CERN DD/EE/80-5, DD/US/87 (1982)

[13] NAG Graphical Supplement
NAG FORTRAN Library Manual - Mark 9
(NAG Central Office, Oxford, 1982)