Graphics and GKS at the UK Meteorological Office

C T Little
U.K. Meteorological Office
Bracknell, Berkshire, U.K.

Abstract

The current complexity of graphical systems at the UK Meteorological Office are described very briefly, and the necessity for standardisation discussed. The reasons for choosing ISO standards are described as are the current status of their implementation, and future plans.

1. Introduction

Early in the 1970s, the UK Met Office had only one mainframe, an IBM 360/195, and only two graphics devices: a Calcomp microfilm plotter, controlled by offline magnetic tape, and an online, channel-connected, direct-beam-refresh vector graphics display, an IBM 2250. Both devices were driven by very similar software packages (Calcomp Electronic HCBS and IBM GSP respectively).

When other vector devices were added during the 1970s, such as offline Calcomp 1136 pen plotters, a version of the HCBS became the de facto standard programming interface. It was used with non-Calcomp devices, and even with raster devices. The main features of this programming interface were:

a) device coordinates,
b) current position,
c) scaling and origin shift,
d) pen number.

The graphical primitives were:

a) single line vectors,
b) non-centred symbol strings,
c) centred marker symbols.

By the early 1980s, there was a CDC Cyber 205 supercomputer, two IBM mainframes supporting interactive colour raster terminals, storage tube terminals, microfilm plotters, and offline pen plotters and many minicomputers with a wide variety of graphics devices attached. The Calcomp based programming interface was now inadequate because it did not have:

a) device independant coordinates,
b) colour,
c) input facilities,
d) device independent picture storage.

There was also a serious software maintenance problem with the many systems being interconnected and numerous conversion packages being written to give compatability between

Calcomp-based applications and various proprietary, device specific, software packages, such as IBM's GASP and GDDM.

2. Initial GKS Experiences

In 1983, an early version of the Queen Mary College GKS implementation (Version 7.0) was obtained and device interfaces for the Calcomp microfilm plotters and IBM GDDM-based terminals were written. Application software for a research project, involving interactive graphical user interfaces and device independence, was developed. This involved the overlaying of complex pictures of vector graphics onto radar images and eventually satellite images. The software ran on the IBM mainframe and IBM 3279 low resolution colour terminals with hard copy directed to a colour matrix printer or monochromatic microfilm. The software was then ported to an IBM 5080 to use higher resolution, more colours and better interactivity. A very simple driver for the 5080 had to be written.

The relative ease with which this GKS-based application software was ported, and the difficulty of writing a 5080 driver, led to the adoption of an organisation-wide strategy of using GKS for new graphical applications, and using a commercial implementation to support the wide range of machines and devices in the Met Office.

GKS was adopted because:

a) It was about to become an International Standard, and was machine, device and software vendor independent.

b) It was a well defined programming interface/language on which to sit application layers (some proprietary programming packages had a mixture of graphical programming routines and application routines).

c) Most proprietary, and 'Core', packages had 3D functionality, and this is unnecessary for most meteorological applications.

d) GKS seemed to have technical advantages over 'Core' (e.g. bundles, no current position).

The GKS implementation from QMC was inadequate at the time because:

a) it was Version 7.0, not 7.4, i.e. it conformed to an earlier draft of the standard, not the standard itself,

b) it was incomplete (some input and most inquiry functions were missing,

c) numerous device drivers would have to be written,

d) there would be inadequate maintenance support for Met. Office purposes.

3. GKS Implementations

In 1985, criteria were drawn up for the selection of appropriate GKS implementations. These were:

a) Availability from one vendor for both IBM and Dec machines, for ease of administration and maintenance and to ensure transportability of applications programs and metafiles.

b) Support for a very wide range of devices, in particular:
   IBM 5080 display terminals,
   IBM 3279 display terminals,
   Ramtek 9460 display terminals,
   Sigmex 7000 display terminals,
   Calcomp pen plotters.

c) Support for GKS Appendix E Metafiles, as opposed to proprietary metafiles.

d) Suitable maintenance and support arrangements (e.g. hot-line support available in Europe rather than the USA).

e) Availability on other host computers, in particular:
   IBM PC,
   IBM 3270PC-AT/GX,
   Dec PDP11.

The first criterion reduced the known field of implementations of nearly thirty down to about ten. The second criterion eliminated all the remaining! However, four implementations were available that claimed to support IBM 5080s, and one that supported Sigmex 7000s. These implementations were all from hardware independent suppliers.

The Sigmex 7000s were supported by the Rutherford Level 1b implementation, GKS-UK, and it was installed on the two Dec Vax11/750s to which these were connected.

The remaining four, commercial, implementations were all tested during 30 day free trials, and benchmarks were run. The benchmarks were:

a) applications to check visually the picture produced,

b) applications that wrote a series of increasingly larger metafiles.

The latter were felt to be realistic tests. The applications contoured complex data on a series of larger and larger rectangular data grids using standard production routines. The Met. Office produces many complex pictures every day, and many have to be produced overnight in batch. The pictures had many polylines, with small amounts of polymarkers and text, and were a typical of meteorological applications.

4. GKS Benchmark Experiences

The process of testing, on IBM and Dec machines, took over one year. Some of the difficulties encountered porting the benchmarks between the five implementations (the four on test, and the original early QMC version) were:

a) whether default primitive attributes were bundled or individual,

b) non-availability of pre-defined bundles on some implementations,

c) adherence to earlier Fortran bindings and versions of GKS,

d) use of specific connection identifiers to indicate variations in workstation type,

e) restriction to one active output workstation,

f) inability to optionally de-activate Metafile Out and Workstation Independent Segment Store workstations,

g) undocumented limits on the sizes of cell arrays, polylines, etc,

h) discrepancies from GKS Appendix E Metafile (e.g. 40 byte first record) or gross inefficiencies (e.g. one item per 80 byte record),

Other problems encountered, but not specifically concerning GKS, were:

a) plain bugs,

b) discrepancies between documentation and software,

c) inadequate installation documentation,

d) inadequate support,

e) non-availability of required drivers,

f) late delivery.

Figure 1 shows some of the benchmark results. In this case the metafiles were Appendix E binary metafiles for the GKS implementations, and device specific formats for the device specific packages.

5. General GKS Experiences

Experience with the QMC GKS, the tested implementations, and the ones now in use (GKS-UK and GKSGRAL) have produced the following list of opinions:

a) As can be seen from the benchmark results, and also found in practice, GKS based applications can take two to three times longer (in CPU time) than the equivalent application using device specific software. However, the applications were originally device specific, and 'grafted' onto GKS in a simple way, and many GKS functions are being duplicated in the application code, hence these results may be unduly pessimistic. There are indications that, using an appropriate device, such as a sophisticated graphics terminal, the higher level functionality of GKS can produce savings, reducing the cost to be more similar to that expected with device specific software (e.g. segments allow coastlines and overlays to be generated once only and then reused). A factor of two increase in CPU time is only just acceptable for some applications, and any increase, without any other advantages, is unacceptable for time-critical production codes. The applications are basically all graphical, with non-graphical calculations usually done separately on a supercomputer. It is not envisaged that the overhead of GKS software compared to device specific applications would decrease significantly until many devices with GKS or the Computer Graphics Interface, CGI, in firmware are available.

b) Internal limitations of implementations stopped the display of large cell arrays (eg 256 x 256). A number of applications were modified to use fill areas instead.

c) For very large cell arrays (eg 512 x 512) a one-to-one mapping onto physical pixels is desired for efficiency and accuracy of the displayed picture. This can be done by:

1) a cell array or GDP such that the world coordinates/normalisation transformation maps onto the physical pixels. There are efficiency savings only if the implementation can 'sense' this (useful for high resolution satellite imagery),

2) bypassing GKS and writing a bit map to the device, and calculating the world coordinates of the corners.

3) use fill areas, but with efficiency problems as the size of each fill area approaches that of one pixel (useful for lower resolution radar imagery),

d) When a display with separately addressable bit planes is used, it is very efficient and natural to store an image in one set of planes, and overlays in other planes, using the colour lookup table to control visibility of each layer. The overlays and underlying image are accurately registered with each other using some world coordinate system. GKS does not fit this conceptual model of a workstation other than by having two or more workstations, assigned to the groups of bit planes, each with identical workstation windows and viewports.

e) When large cell arrays (or GDP bit maps) are displayed, standard Fortran requires these to be stored in the application program in integers (e.g. 8 bit image colour indices

stored in 32 bit integers). A solution is to augment Fortran 77 with shorter integers, or perhaps bit or byte data types.

f) Two different views of a single object (possibly being updated) cannot be displayed without explicitly copying the object and updating both copies (e.g. a small picture plus a large zoomed portion of it).

g) A large number of application specific symbols and markers are required, but there is no standard method of supplying the requisite font information to the GKS implementations.

h) Because of the disparate devices used (e.g. a high resolution monochromatic microfilm plotter and a low resolution colour raster terminal), bundled attributes are found to be the best way of providing device independent application software, providing the bundles are initially defined in the implementation.

i) Numerous programmers, of varying quality, have been taught GKS. One of the most problematic areas was the teaching of segment manipulation using the Workstation Independent Segment Store, the Copy, Associate, and Insert functions and their effects on the displayed output.

j) The least used parts of GKS have been the query functions and the sophisticated text control functions.

k) It has proved impossible to get device drivers for all the devices available at the UK Met. Office from one, or even two, vendors. It is hoped that the finalisation of CGI as an International Standard will make it possible to buy drivers and GKS from different vendors and have them work together.

6. Metafile Requirements

These are:

a) the storage and transmission of large numbers of complex pictures, hence compactness and efficiency are of prime importance,

b) the control of the appearance of a stored picture at view/interpret time (i.e. to allow a picture to be exported to, say, a TV company, who would enhance the appearence, but without altering the geometric information),

c) within a metafile, the storage of seperate layers of a picture (e.g. map background, sets of contour overlays, satellite image underlays) to allow optional combining at view/interpret time,

d) a mechanism for accessing a specific picture rapidly in a large metafile is required (e.g. to allow a user to skip forward or backwards a specified number of pictures, but without having to view the intervening ones). This would

allow a machine independent graphical spools to be constructed, for use in the same way that spools are used now for alphanumeric output.

7. Summary

GKS is an appropriate standard for the UK Meteorological Office. However, many implementation dependant variations exist in implementations which does not enable portable software to be produced very easily. Efficiency is an important issue until the majority of devices incorporate GKS or CGI functionality in firmware or hardware. The display of large images with GKS graphical overlays is a problem and cannot be efficiently tackled within GKS.

Because the efficiency and compactness of metafiles is important, the GKS Appendix E metafile is inadequate and the Computer Graphics Metafile is preferred. Hence GKS and CGM must be compatible with each other as much as possible.

CGI is not adequate or appropriate as an application programming interface because multiple devices must be driven by single applications. It is hoped that GKS and CGI will be compatible with each other so that CGI devices can be driven by GKS.

The Programmer's Hierarchical Interactive Graphical System, PHIGS, has too much functionality for current requirements and GKS-3D is not required at present either.

In the short term, CGM will be installed in the UK Meteorological Office to replace GKSM, and the library of meteorological symbols made available via GKS. GKS will also be extended to support bit maps for the efficient display of imagery.

At present, the user interfaces of interactive applications are relatively unsophisticated and application dependent. In the next few years, application, device and host independent standardised user interfaces will be required. In particular, alphanumeric screen menus, used in conjunction with a graphical display, need standardisation. Also the standardisation of graphical applications writing to a simple window on a sophisticated device is required.

Figure 1

Benchmark Timings Writing a Binary Metafile