stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IN 17/79 FEBRUARI

P.J.W. TEN HAGEN & F.R.A. HOPGOOD

TOWARDS COMPATIBLE GRAPHIC STANDARDS

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

ISO/TC97/SC5/WG2-Graphics

Editorial Board Meeting,

Amsterdam, february 8-10,1979.

The Editorial Board implements the recommendation by the above ISO working group, to establish a direct cooperation between all national groups currently developing a graphics standard based on the GSPC CORE report. On request of the national groups concerned (e.g. ANSI and DIN), the ISO working group provides a neutral chair for the board, as well as a comparison between both national efforts by members of the ISO working group.

The comparison of the national efforts is based on two documents often referred to as the CORE and GKS respectively. These documents are:

- Status Report of the Graphic Standards Planning Committee of ACM/SIGGRAPH, Part II: General Methodology and Proposed Standard, Computer Graphics, Volume 11, no 3, fall 1977.

- Proposal of Standard DIN 00 66 252, Information Processing, Graphical Kernel System, Functional Description, Preliminary Version.

Statements in the following attributed to ANSI/GSPC or DIN or regarding the philosophy of the CORE or GKS are not official positions but represent the opinions of individual technical experts who are well acquainted with the national efforts involved.

Contents

List of Participants

Ralph Eckert (DIN), TH Darmstadt-FB Informatik, FG Graphische Datenverarbeitung, Steubenplatz 12, Darmstadt,BRD.

Ir. Johan Ero (NNI), Secretary to the meeting, Philips Dep. ISA-VN523, Eindhoven, Netherlands.

Drs. Paul J. W. ten Hagen (NNI), co-chairman, Mathematical Centre, 2nd Boerhaavestraat 49, Amsterdam, The Netherlands.

Prof. F.(Bob) R.A. Hopgood, (BSI), co-chairman, SRC, Rutherford Laboratory, Chilton Didcot, Oxfordshire OX11 0QX, Great Britain.

Dr. Klaus Kansy (DIN), Gesellschaft fuer Mathematik und Datenverarbeitung mbH Bonn, Postfach 1240, 5205 St. Augustin, BRD.

Dr James C. Mitchener (ANSI), Intermetrics Inc., 701 Concord Avenue, Cambridge, Massachusetts 02138, USA.

Agenda

0. Opening

1. Main principle differences between GSPC CORE and DIN GKS

1.1 3D - 2D

1.2 Storage Philosophy

1.3 Inquiry Strategy

1.4 General Comments on GKS by Michener

2. New developments around GSPC CORE

3. Comparing Functional Groups

3.1 Output Primitives

3.2 Segmentation and Naming

3.3 Attributes

3.4 Viewing Transformations

3.5 Input Primitives

3.6 Control

4. Future work of the board

5. Final technical remarks

0. Opening

Paul ten Hagen welcomes all participants.

The definition of the work of the board for this meeting is, to compare the two national efforts towards a graphics standard, namely GSPC CORE report and DIN GKS functional description.

Statements in the following attributed to ANSI/GSPC or DIN or regarding the philosophy of the CORE or GKS are not official positions but represent the opinions of individual technical experts who are well acquainted with the national efforts involved.

The result of the comparison will be to list the differences and to provide ways to resolve them. The latter will be formulated in the form of recommendations. A recommendation will always address one or both functional descriptions. A change of a description according to the recommendation will bring them closer together.

The meeting will also identify areas which need further clarification.

1. Main Differences

Main differences are discussed first because they affect more than one functional group. The result should clear the ground for resolving differences on the functional level.

1.1 3D - 2D

The GSPC CORE proposal supports 3D-output whereas GKS supports only 2D.

Comments by Michener: Allthough a majority of people requires 2D only, the part that wants 3D to be supported is very rapidly growing and is expected to become the majority within a few years.

In the CORE, the relationship between 2D and 3D is that 2D is considered a special case of 3D. The omitted z-coordinate in the 2D-case is assumed to be the z-coordinate of CP. 2D output does not affect z-coordinate of CP.

GSPC feels that 3D should become part of the standard now in order to prevent people developing their own 3D extension on top of a 2D CORE.

Comments by Kansy/Eckert: The reason for omitting 3D is mainly lack of manpower. The DIN group has to come up with a proposed standard in a very short time. Therefore they have left out a few functional groups for which only a small minority of their users has an immediate need. They will certainly add 3D when time permits.

The important question that remains now is: Will the future 3D - extension of GKS be consistent with the current 3D CORE proposal?

All agree that 2D should be considered as a special case of 3D.

There remain a few questions however:

1. Is it allowed to have 2D and 3D primitives mixed in one segment?

It is allowed in the CORE, whereas the current opinion in DIN tends to forbid it.

2. Does having CP impact the way 2D is a special case

of 3D?

It does in the CORE. In view of the fact that GKS might not use CP it could not define 2D-3D relation in the same way.

The board feels that the CORE wants to use CP for introducing convenience rather than basic semantics.

RECOMMENDATION 1 :

The CORE should try to avoid using CP for defining an important item like 2D-3D relation.

RECOMMENDATION 2 :

In view of the fact that the 2D-3D relation can be clearly defined (2D is special case of 3D), there should be an optional configuration of the CORE having 2D only.

1.2 Storage Philosophy

Eckert/Kansy: GKS distinguishes graphical objects and pictures. They correspond to CORE segments in different stages along the viewing pipeline.

Objects are stored in the so-called GKS-file as untransformed and non-clipped segments.

Pictures are stored at the workstation as transformed and clipped segments in a device dependent or device independent display file. Which one, is defined by the implementer only.

A workstation is allowed to refuse to store segments. This is similar to the fact that in the CORE an output level2 device driver does not support retained output.

It is established that the concept of CORE segment is the same as the concept of a segment "on" a workstation (picture). The GKS-file is a form of storage not supported by the CORE.

Associated with the GKS-file are the following explicit or implicit functions:

1. Allow creation of segments on different workstations from one sequence of primitive invocations. This function is needed in GKS because there can be only one workstation active at the time.

2. Insert segments from GKS-file into the open segment.

3. Deliver segment on record for long term storage.

The following remarks on GKS-file turn out to be important:

A GKS-file is a high level facility.

A GKS-file is erased at the end of a session.

Processing of GKS-files requires inspection of GKS-file segments contents.

GKS-file contents is completely device independent.

An object in the GKS-file has a symbolic form of attribute whose meaning is determined only when the object is inserted in a segment on a workstation.

An object in the GKS-file cannot specify what line style, line width, color or intensity primitives have.

The GSPC CORE envisages a file facility on top of the CORE, i.e. a high level facility also.

The argument in favor of GKS-file saying that "outputting the same picture to a second device driver requires a complete regeneration by the application program" is countered by the fact that the application program might do that very well from an application-specific representation. In some application programs, this will not be less efficient than using GKS-file capability. The GKS-file functions INSERT,DELIVER,GENERATE and the inspection of segments are compatible with a CORE file module for short term or long term filing.

Having a temporary file for output to multiple workstations is considered a different form of filing.

RECOMMENDATION 3:

Change the GKS-file terminology:

GKS-file -> temporary file or object structure.

Say records are stored in or read from files by the application program.

RECOMMENDATION 4:

Do not use the GKS-file for supporting multiple devices. Multiple devices should already be present on levels below the GKS-file level.

RECOMMENDATION 5:

Define a level and/or module structure for GKS.

It turns out that the concept of levels is a very useful means to isolate and resolve differences.

The support of multiple devces without temporary filing could be done at one workstation for the following reasons:

A workstation can have more than one output device. Rather than giving the operator an external device switch, one could put the switch under program control by some function. This is the more disirable for input (and in fact already the case).

The editorial board feels that DIN can maintain the requirement of one workstation active at the time. Transfer between workstations has to be done by the file system but it now has a different meaning.

RECOMMENDATION 6:

Existence of a file system on top of the segment system should not affect the way the segment system underneath works. The segment system should include all facilities desirable at that level. The introduction of a file system on top of the CORE should not change application program structure.

RECOMMENDATION 7:

DIN should investigate the possibility of supporting multiple view surfaces by the segment facility.

At a workstation one can change the pen representation for a given pen number. The binding between pen number and pen representation is done at a very late stage. In this way a picture may change after UPDATE. This type of facility requires a lot of overhead, e.g. a device independent display file. This type of change is not possible in the CORE.

There are three reasons for having the pen number construct.

1. To have application program control over different visual effects on different devices.

2. To permit library routines to easily select attributes defined by the calling program.

3. To dynamically modify existing segments on a workstation by changing the pen representation.

The editorial board feels the first reason is most important because a given device might not support certain attributes.

A way to decrease the difference between CORE and GKS would be to apply the pen number concept in this situation only, and therefore not to change attributes by it.

1.3 Inquiry Strategy

DIN feels that if all possible inquiry functions are allowed, there would be of the order of one hundred functions. This is too high and some method must be defined which limits the number of inquiry functions. The GKS system has a number of independent datastructures which at different times are accessible from the applications program. The GKS philosophy had been to try and encapsulate the information that could be inquired so, that it was unavailable when it was not required.

The GSPC approach was that it was impossible to define states where the information in the various states lists might not be legitimately required even if only to store for later use, or for debugging. Some editorial board discussion attempted to see if it was possible to restrict inquiry to information that one can use immediately.

RECOMMENDATION 8:

In general all status information should be available irrespective of the current use.

Any errorhandling provided cannot use the inquiry functions but must have immediate "error free" acces to the datastructures. Otherwise it is possible to generate recursive calls.

RECOMMENDATION 9:

GKS should define a set of Inquiry Functions. They must be part of the kernel proposal and not left to the language dependent level.

RECOMMENDATION 10:

GSPC should provide a clearer definition of functions which cause major state changes and to make more explicit the states and/or flags which constrain what is allowed in various situations.

The categories of information open for inquiry are, but for image transformations of a segment, similar in both proposals.

1.4 GKS Philosophy

Some discussion took place concerning the GKS introduction (section 3.1), which describes GKS philosphy. This suggests that the layers described there were all of non-zero thickness. This implied that the Application Oriented Layer could not call GKS primitives directly, even if there was a 1-1 equivalence between these and the language dependent layer. It became clear that the GKS proposals were not so rigid in this respect as they seemed.

2. New Developments Around GSPC CORE

Jim Michener gave a synopsis of GSPC changes.

1. Levels have now been devided into input levels and output levels. In theory, any input level can be taken with any output level allthough they are not completely orthogonal, because one cannot have pick input (input level 3) without segments (output level3).

2. A new GSPC 79 was due out on january 22 and is probably currently in the mail. Comments are invited before the Boulder meeting.

3. POLYMARKER has been added as an output primitive and the marker type is now an attribute for both MARKER and POLYMARKER.

4. Text attributes are now more complex. It is possible to specify text output to go from left-right, right-left, up-down and down-up to allow for human (natural) languages other than English. Variable spaced character fonts will be allowed. Justification will also be available as an option (begin, end and centered).

5. The non-retained segments have been removed. Instead it is possible to call output primitives, attributes and define transformations outside of segments. This was provided for plotter users etc.. An alternative approach which was not accepted, was not to have default settings on CREATE SEGMENT. This situation may well change.

6. A much more precise definition of attribute values has been specified. For example, line styles 0 to 9 will be used for hardware line styles and line styles above 9 define the precise dash-dot pattern (similar to PLOT10). Also more precise are defined the range of values for segment names and for pick-ids.

7. A set of 10 standard markers has been defined.

8. Image transformation values are now completely specified in one function call. The motivation is to save multiple passes through the segment file.

9. Input devices must now be initialized before they are enabled.

10. Additional inquiry functions have been provided, as has a function which sets all viewing parameters at once.

11. Two types of picture change control have been included that affect the behaviour of primitive creation, seg-

ment attribute settings, segment deletions, and NEW FRAME.

The first one turns off immediate visibility so the display can be arbitrarily out of date, similar to I/O buffering. CLOSE SEGMENT has no effect.

The second type mandates four alternatives for batching of updates on a view surface. The implementation can choose which one:

- 1. Like a refreshed device - state of batching ignored.

- 2. Like a storage tube device - deletion is saved up but additive changes go through.

- 3. All retained segment operations are delayed but non-retained output is immediate.

- 4. All changes are saved. An example might be a device where overhead for immediate output is large. For example, line printer output with raster conversion.

12. Input will now allow different kinds of echoing to be specified. For example, rubber banding echo for locator input.

13. The extensions sub-committee is considering raster displays, colour and pseudo devices (file transfer but with segment information lost).

3. Comparing Functional Groups

3.1 Output Primitives

The main differences to be discussed were:

1. To have CP or not
2. 3D output or not
3. DRAW/INSERT high level primitives

The GSPC use of CP is purely a syntactic abbreviation. Setting CP does not create output primitives, it only influences how they are produced. Thus it is just a convenience. The major motivations are that a lot of systems have it and it reduces the number of parameters in future calls. It also allows the specification of the output primitives using calls of functions which have relative coordinates.

A major problem with the GSPC definition of CP is that it is used implicitly in the mixing of 2D and 3D output primitives. For example,

```
MOVE-ABS-3 (X,Y,Z)

LINE-ABS-2 (X,Y)
```

The second primitive uses the Z-coordinate of the first.

DIN agrees that CP is current practice. However, CP has a number of severe conceptual consequences. The GSPC definition has to mention CP frequently. It is equivalent to the use of an accumulator in assembly language programming. The DIN view is that the output primitives should be on a high level. It would not be acceptable to DIN to build a language dependent level on top of the GKS system which included CP as construct. GKS defines primitives at the POLY-level. DIN feels that even a LINE primitive is unnecessary. In most applications, data values are not generated inside the program but are input from device and so the user does not need to write cumbersome seven-line POLYLINE calls very often,e.g.

```
A(1) = 5; A(2) = 6

A(3) = 7; A(4) = 5
```

```
A(5) = 6; A(6) = 8

CALL POLYLINE(A)
```

During a long discussion the editorial board tried to resolve the opposite approaches. Certain mixing of output calls in the GSPC proposal were not easily transformed into equivalent GKS calls. For example:

```
LINE-ABS-2(X,Y)

TEXT ('ABCDE')

LINE-ABS-2(A,B)
```

If GSPC was to not allow such uses of CP, it would be possible to translate GKS primitives into GSPC and vice versa.

The following was agreed:

RECOMMENDATION 11:

The concept of CP is a major fundamental difference between the two proposals which needs to be considered urgently.

A possible way of resolving the current impossibility to allow each system to be specified in terms of the other would be:

RECOMMENDATION 12:

The CORE should not change the CP for text output. Instead an inquiry function should be provided which gives the last end-of-text.

RECOMMENDATION 13:

GKS should add an INQUIRY function which gives the last end-of-text.

Some discussion of GKS DRAW compared with GSPC ESCAPE ensued.

RECOMMENDATION 14:

DRAW (GKS) and ESCAPE (CORE) are higher-level primitives which should not appear in the lowest level of the system definition.

RECOMMENDATION 15:

At some stage, there will be a need to standardise on the language level representations such as how an array of points is stored. DIN and GSPC should adopt a common array definition strategy and function parametrisation strategy - particularly for FORTRAN.

For example, does POLYLINE-3 take an N by 3 array or three N by 1 arrays?

Allthough it was suggested that GKS should introduce LINE and MARKER primitives, it was felt that this was neither a major nor a significant difference (unlike some of the other issues that had been raised).

3.2 Segments

Major issues discussed were:

1. Retained/Non-retained Output

2. No RENAME command in GKS

3. Deletion of segments on more than one workstation

In view of the discussion on storage philosophy this section deals with segments(pictures) only, and not with files(objects).

The DIN view is that non-retained output outside a segment is a new concept which is not necessarily bad. However, it will have to be discussed within DIN first. The same can be said of attribute setting outside segments.

GSPC feels that the CORE philosophy of having CREATE/CLOSE force default attribute settings makes the CORE easier to understand. Also it is good practise as it means that the definition of a segment cannot be changed by some global setting lexically remote from the segment definition. Therefore it is much easier to read existing application programs. The new CORE definition means that the CLOSE SEGMENT function in effect is an implicit 'open non-retained' function. Attributes for the non-retained output are not saved across the segment definition.

There is the possibility that the next version of the GSPC CORE allows viewing transformations to be changed within segments.

There is also a possibility that non-retained segments will "return" but that setting defaults at CREATE SEGMENT will be eliminated.

There was an agreement that no recommendation could be made at this stage as both systems were in a state of change.

RECOMMENDATION 16:

The RENAME SEGMENT function should be included in GKS.

There was a lengthy discussion over segment naming. Allthough GKS segment names are stored centrally to check that the name is not reused, most of the information concerning the segment is stored on the workstation. A segment could only be on one workstation.

The recommendations agreed upon were:

RECOMMENDATION 17:

GSPC should change the name of SELECT and DESELECT as normal usage implies a total deselection which does not in fact take place.

Selecting a device for output in GSPC CORE is independent from segment management, e.g. a segment can be deleted on a non-selected surface.

RECOMMENDATION 18:

In GKS and CORE it should be possible as an option, not a mandate, to output primitives to more than one view surface with different pen representations on each device. Such a facility should be available without using the GKS-file system.

This implies in GKS either workstations should be simplified and allow multiple workstations or expand the facilities of a workstation to include multiple view surfaces within one workstation.

3.2.1 Naming

The DIN position is that 2-level naming should imply 2-level modifying. It is only used for menues and this is better done by using a special CHOICE input primitive handled by the driver.

The GSPC position does not worry about consistency of naming and modification. The second level is only provided for PICK input. They see the PICK-ID as being used in two different ways:

1. Light button menues

2. Scatter plot or graphical entity to be picked (Operator tells system to delete that point over there.)

The GKS approach does not allow menues to be made up of graphical entities unless they are defined at the driver level. Thus the application program cannot define a menue in one segment via the standard.

There is an additional argument for two level naming (which DIN rejects) saying that segment overhead is much greater than the equivalent PICK-ID overhead.

There is also the experience from GINO user groups which asked for two level naming.

RECOMMENDATION 19:

There are valid reasons for a second level of naming and DIN should reconsider. The menue argument is independent of the second level of naming.

Two other problems concerning segments were:

1. CORE permits the open segment to be manipulated - GKS does not

2. Image transformations should not alter the segment contents. That is, image transformations are not accumulative in the GSPC view.

GSPC's view on open segment manipulation is that it should be allowed to pause in the middle of an open segment and allow the segment name to be available. In particular, it should be possible to change the visibility of the open segment.

GKS's view is that one should restrict the use of functions for security and good programming practice. Application programs should not be allowed to change the open segment without first closing it.

In the discussion it became clear that GKS asserted

that invisible segments could be sent to the workstation and yet that there was no mechanism for doing it.

RECOMMENDATION 20:

That DIN implements the GKS statement that "Invisible segments may be sent to the workstation" (page 7). There needs to be a capability for creating invisible segments and, all segment attributes should be subject to change in an open segment.

GSPC suggests that it should be possible to change segment attributes (visibility etc.) without having to activate a workstation or view surface.

Next image transformations are discussed.

If one accumulates matrices one gets round off errors. A pure rotation will eventually also include some scale function. Therefore it should be possible to replace as well as accumulate image transformations. This necessitates keeping transformed and untransformed segments in some cases. CORE also provides an inquiry for the current image transformation. If one only keeps a composite of the values applied sofar, it is impossible to read the values back.

The DIN view is that one should transform the transformed data if it is being done by software. Motivation is to reduce the space overhead.

This argument may be weakened by having to keep a device independent file, anyway, for pen attribute capabilities.

No recommendations were made as this area needs further clarification.

The editorial board discussed whether scaling should take place before rotation takes place in image transformations. There appear to be some arguments for both choices. The full pros and cons do not seem to have been written down.

RECOMMENDATION 21:

A clear statement of the pros and cons for:

1. scale-rotate

2. rotate-scale

3. scale-rotate-scale

should be set out by both GSPC and DIN.

3.3 Attributes

The major differences with respect to attributes are the following:

1. GKS has no segment type for image transformations. It has a store mode instead of the retained/non-retained attribute.

2. Pickid attribute (already discussed as part of segments).

3. GKS has pen number concept instead of corresponding modal attributes.

4. There is no charplane function as text attribute in GKS. Neither of the additional attributes in the revision of the CORE is present in GKS.

5. It is not certain whether set visibility, set highlighting and image transform have an UPDATE side effect in GKS, like the NEW FRAME effect in the CORE. Visibility, highlighting and detctability are not independent in GKS.

Discussion of 5.:

GKS gives an error when a segment is invisible and detctability or highlighting are set. After discussing several examples the general feeling is that such settings ought to be independent.

RECOMMENDATION 22:

GKS should allow SET VISIBILITY, SET DETECTABILITY and SET HIGHLIGHTING independently.

As an addition to the earlier discussion about pen attribute the following remarks are made:

Pen attribute of GKS meets the requirement of having "different attributes on different devices".

If the penrepresentations per pen number were not allowed to change dynamically (i.e. cause segment modifica-

tions), there would be only a minor problem.

Could GKS provide motivation and case for change of pen representation causing a segment(picture) modification?

Dynamic change of pen representation is particularly difficult for workstations that store segments.

A change of pen may have the side effect of updating.

DIN remarks:

Pen attribute will become more interesting in combination with GKS-file.

DIN would like to see pen attribute as an option (i.e. not present in a basic system).

It is desirable to have the display of a segment on different devices appear to be different. This difference should be under application program control.

The GKS function UPDATE is very similar to the pair of CORE functions END-BATCH-OF-UPDATES / BEGIN-BATCH-OF-UPDATES.

Immediate update as a side effect is, in GKS, left to the implementer.

Michener: complaints about the GSPC 77 definition that "batching of updates is implementation dependent" were quite bitter. I strongly recommend that DIN define a number of mandatory, precisely specified ways of possible implementations of batching.

DIN feels that reading input should force an implicit picture update. GSPC has explicitly rejected this, because of the example of a digitiser, used at full speed, being significantly ahead of the graphical output.

GKS should set out as clearly as possible what is intended in terms of what attribute changes cause a picture modification.

Next segment type is discussed.

GKS has no static segment attribute specifying image transformability. The following comments are made.

DIN: Transformability is a feature of the workstation.

In most applications either all segments are transformable or none are. Therefore transformability should not be a segment attribute. Rather one should specify at the global level something like "no 3D transformations are used".

GSPC: The properties of refined transformability fit within the attribute scheme. Not all segments should pay the overhead of storing information (e.g. 3D screen coordinates if only a few segments use it).

It turns out that an important question must be answered. Resolving the CORE/GKS difference very much depends on the answer:

To GSPC:

What happens when a certain view surface cannot support certain segment types?

What happens when you try to use it?

Two final remarks conclude the discussion on attributes:

The attribute refinements in GKS (linestyles,color etc.) are expected to be the same as are given now in the CORE revision.

GKS should note that GSPC has changed its text definitions which now makes the two systems incompatible.

3.4 Viewing Transformations

The major issues were:

1. Should there be a clipping rectangle seperate from the window (for defining the window/viewport mapping)?

2. Should the CORE VIEW-UP-2 function be implemented in GKS for compatibility with 3D?

3. Is the mapping from NDC to device coordinates the same in both systems?

4. Is the WINDOW specified by points or limits?

Points versus limits:

The decision whether the window is specified by points or limits is purely an explanatory matter. GKS should make it quite clear that the order of the two points does not imply any transformation of the viewspace (reflection etc).

The clipping rectangle:

RECOMMENDATION 23:

GSPC should consider the separation of the clipping rectangle from the window definition.

RECOMMENDATION 24: GKS should change the name of the window.

VIEW-UP-2:

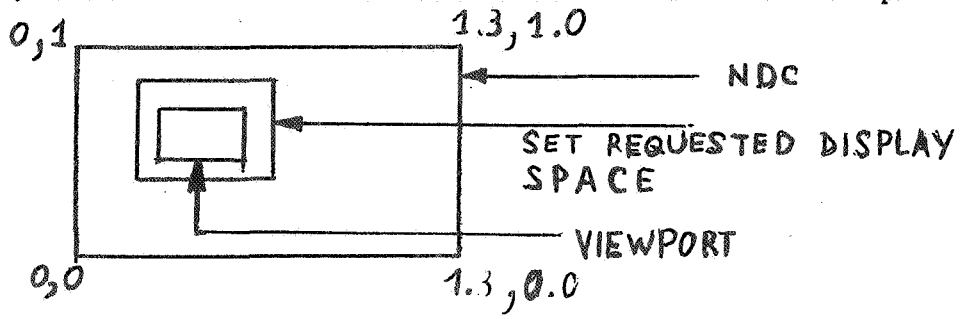
RECOMMENDATION 25:

When GSPC defines a 2D subset, VIEW-UP-2 should not be included in it.

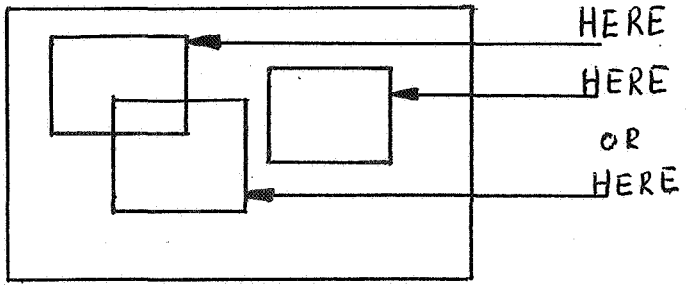
It was ascertained that both GKS and CORE had inquiry functions which allowed the actual dimension of the view surface to be obtained.

Mapping from NDC to Device coordinates:

The GKS function SET-REQUESTED-DISPLAY-SPACE is really only provided for efficient use of the plotting area for multiple plots on a drum plotter. GKS binds the NDC to the device independent of the application program. The origin is in the lower left hand corner and 1.0 corresponds to the smaller device dimension. SET REQUESTED DISPLAY SPACE defines a rectangle in the NDC space that will be used for the plot. An error occurs if the requested rectangle is outside NDC space. All viewports(and the "images" of the clipping rectangles) must be defined inside this area. For example:



The driver supporting the plotter has complete freedom to locate the display area for the plot at any position on the plotter surface to save paper etc., e.g.:



Thus with respect to controlling a plotter, the limits of the requested display space only defined the area to be used. Consequently, SET REQUESTED DISPLAY SPACE could equally well define the area in the lower left hand corner. Defining it aywhere else is just confusing.

On a device other than a plotter, SET REQUESTED DISPLAY SPACE is ignored. Michener suggested that it would be better to map the requested space to the full display screen.

RECOMMENDATION 26:

SET REQUESTED DISPLAY SPACE should define an area in the lower left hand corner(P parameter is the origin). It should also force a call of REQUEST NEW DISPLAY SPACE.

The following question was raised:

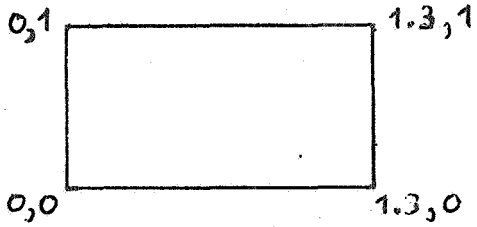
If clipping is turned off, should devices be expected to provide clipping at the boundary of the view area?

The DIN approach is that hardware devices should do this and, if they do not it will be provided in the driver. The GSPC approach is that one need only to protect an unintelligent device from damaging itself.

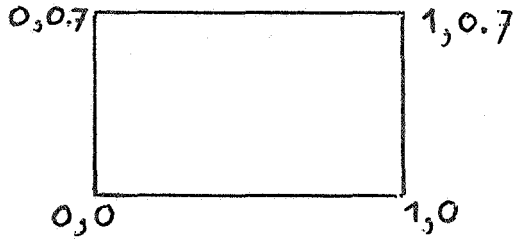
The two approaches to defining NDC space are as follows:

1. GKS insists that the aspect ratio of NDC space is device dependent, that the unit square is visible on the device and that it appears in the lower left corner. For ex-

ample a horizontal Tektronix would have:



2. CORE insists that the aspect ratio of NDC space is application dependent with a default of 1:1, that the limits of the view area must both be less than or equal to 1. Thus if a 1 by 0.7 NDC-space is requested one has:



In this way, the CORE achieves greater application portability than GKS. However it is defined, the view area is portable and visible on all devices. The CORE does not constrain the origin to be the lower left corner.

RECOMMENDATION 27:

There appears to be no good reason why the choice of NDC is different in the two systems. The GSPC approach seems to have some advantages. GKS should reconsider and explain the advantages, if any, of this approach over GSPC.

3.5 Input Primitives

There is a very big difference between GKS and CORE with respect to input.

GKS offers a very minimal set of capabilities. GKS defines 5 logical input devices which may be at a workstation. In contrast CORE defines 5 classes of logical input devices.

GKS defines five read functions like the FORTRAN read statement (without the format capability, however) to obtain data from one of the five logical devices.

Because of the apparently new approach Kansy/Eckert are asked to give the general idea behind this, as well as the reason for not following the CORE.

DIN comments: In the previous revision of GKS, event and sampled input was present. The current proposal is a result of critique on the previous one, by DIN members. It was decided to have a much simpler input first and add a high level module containing sample and event later. The simple version also allowed to interface to a FORTRAN read.

From dicussing the previous draft a number critical comments apply to the CORE as well:

Every logical device should be usable both as sample and as event device. Thus, locator and valuator may also be event and pick, text and button may be sampled. If all input devices may be event drive, no association of button and locator would be necessary. DIN considers such an application of a button not a choice. Furthermore input of sampled devices is not under operator control.

As GKS wants to base itself on read, the time perameter as used in GSPC is not meaningful.

In the CORE there is no adequate definition of a logical input device. The description therefore needs inproving.

Comments from GSPC:

Input is broken down to the lowest common denominator. Natural associations are inquirable or at least known from an "installation manual".

The application program should time out, or prompt in case of await event for a limited amount of time.

There are two sample input strategies available to dev-

ice drivers:

1. Inform the operator that a sampling is required and await his response

2. Return a current value

Input device simulations may use either.

GSPC defines locator and valuator logical devices to be those parts of input mechanisms that determine positions and values. Those parts of input mechanisms that determine instants in time are called event devices.

Discussion:

In both CORE and GKS logical devices are defined by the type of value they deliver. The main difference between the CORE and GKS(revision 2) input proposal regarding what is a logical input device, is that CORE uncouples sample and event devices.

Michener: With GKS, the operator cannot choose between using one device or another (DIN agrees this is sometimes desirable). If the five read functions were replaced by await-event, the operator could be given this capability.

DIN is urged to proceed with plans to define event and sample input as an option of GKS which operates under the read functions to determine whether adopting read will be incompatible with future capabilities in this area.

RECOMMENDATION 28:

GSPC should include an inquiry capability which informs whether a sample request will hang up the application program until a new value is present.

RECOMMENDATION 29:

GSPC should consider sample and event as operating modes for all logical input devices. This mode can be specified at enable time.

Two more points concerning input remain:

1. Locator in GKS returns user coordinates. Locator in GSPC returns ND Coordinates.

2. Input in continuous mode in GKS is very far away from event queue. This is the main reason for reading sets of locator values. The event concept may be different to support over a communication line in case of a non-stand

alone graphics system. This question is not further resolved in view of the previous discussion.

With respect to the problem of a locator returning user coordinates the following comments are made.

From DIN: It is recognised that this may be difficult to maintain in the 3D case. The examples in the last issue of computing surveys however, show only use of locator values mapped back on world coordinates.

It is believed that almost always a locator value must be related to the user space connected with the picture on the screen.

From GSPC: In 3D "backwards viewing" transformation is sometimes ill-defined. When well defined it requires a lot more work than converting physical device coordinates to NDCs. Assuming world coordinates forces the application desiring NDC to do double work. It also is assumed that the "current viewing transformation" is an output concept, not an input concept too.

In case of an asynchronously provided locator value the mapping onto world coordinates can only be performed at dequeuing. The viewing transformation at this time may not be the viewing transformation the operator intended to be used.

A splitscreen technique (simultaneous multiple view of the same object) means that the desired NDC to user map is "piece-wise linear" and is discontinuous at viewport edges.

There are examples where the locator value is intended to be a point in user space.

GSPC and GKS should give their motivations for defining locator result in user space or in NDC space in order of importance.

The following possibility is discussed:

Have both mappings (viewing, locator to user space) independent. Define a conversion between the two. Allow the input mapping to be switched off.

RECOMMENDATION 30: Give in both GKS and GSPC CORE a function which controls whether locator values are returned in NDCs or in some user coordinates. Provide also two functions one of which defines locator to user space mapping as reverse viewing whereas the other defines viewing as the reverse locator mapping.

Final remark on input: The valuator in GKS is not forced in the range [0,1]. It is an error in the report.

3.6 Control

The major differences discussed were:

1. Error handling
2. Batch of updates
3. Levels versus modules
4. Initialisation

Error handling:

GSPC comments: GSPC believes there are only two kinds of errors in a debugged system: GKS type I and III. GKS lists errors of type I and III only. (Type II errors in GKS attempt to save current graphics data). In the CORE there is a distinction between errors and warnings.

GKS comments: Error handling is not a specific graphics issue. Chapter 7 sofar is only a guideline.

It is established that one can standardise which errors to report, but error handling cannot be standardised.

RECOMMENDATION 31:

Errors should be listed and numbered in one place in the document. Also each function should specify which errors must be detected before any status change, and which need not be so detected.

This kind of classification determines some details of an implementation strategy.

In the CORE a user supplied error handler can be called instead of the standard error handler.

RECOMMENDATION 32:

GSPC should establish a seperate list of errors and error numbers.

RECOMMENDATION 33:

A user defined error handler cannot call any GKS or CORE functions. GKS should make this explicit.

With respect to picture change control(UPDATE):

I/O buffering is desirable for having efficient transmission. It causes problems however with immediate visibility.

There are a few device dependencies hidden in the control section:

In GKS a workstation can refuse to store segments.

In CORE an output level 2 driver cannot store segments even in an output level 3 implementation

RECOMMENDATION 34:

Try to define all device dependencies in the control area.

4. Future Work

The report of this editorial board meeting will contain:

- A clear statement of the editorial boards status.

- Minutes and Recommendations.

- A list of areas where further clarification is needed before any recommendations can be made.

- The comments from ISO/TC97/SC5/WG2-members concerning the comparison.

The report should be ready with sufficient number of copies within approximately one week to be distributed amongst ISO-members and correspondents, DIN, ANSI and GSPC people. As a consequence high polishing of the text will not be possible.

An attempt will be made to get as much feed back as possible from the GSPC Boulder meeting as well as the next DIN meeting. In general it is expected that feedback should contain at least:

- Comments on all recommendations, especially containing arguments when rejected.

- All clarifications asked for.

Feedback is requested from ISO/TC97/SC5/WG2-members as well.

The amount, quality and content of the feedback will decide if and when the next meeting of the editorial board will take place.

5. Final Technical Remarks

Finally a number of minor questions are briefly discussed.

Some names need reconsideration: DIN feels that CORE names for some logical devices sound to much like physical devices (keyboard, button). CORE feels that names of logical input devices should sound like input (e.g. "pick" does, "text" does not).

In GKS the word SET in the input functions REQUEST SET OF... is really a sequence or a tuple.

GKS will maintain an explanatory part next to a precise definition.

GSPC suggests rule zero on page 40 (GKS):

All GKS functional capabilities have to be accessible for each laguage dependent layer defined.

Question rule 2 (GKS page 40): Will error report change after renaming?

Michener: Can there be a single standard device driver interface? GKS: This is meant to be an implementation guide line.

State "segment open" is three states really.

Can a workstation arbitrarily refuse any pen representation? DIN: Yes, this is implementation dependent.

Clarification Areas

This list contains a number of undefined or unclear areas in both reports, which are not indicated as part of a Recommendation. Every area is preceded by a page number refering to a page in this report.

It is also indicated which one of the two reports,if not both, is addressed.

(17) (GKS,CORE) What is the strategy of setting attributes and default attributes inside and/or outside segments?

(19) (GKS,CORE) Should image transformations be accumulated,replaced or both?

(21) (GKS) Give motivation and case for change of pen causing picture modification. What attribute changes cause picture modification?

(22) (CORE) What happens if a view surface cannot support a segment type?

(26) (CORE) Give a definition of a logical input device.

(28) (GKS,CORE) Give motivation for locator->NDC->user space mappings, plus order of importance.

Appendix

Comments by ISO/TC97/SC5/WG2-members:

- G. Krammer et al, Hungary
- D.S.H. Rosenthal, U.K.
- R. Williams, U.S.A.
- K.W. Brodlie, U.K.
- BSI DPS13/WG5, U.K.
- J. Encarnacao & P. Wisskirchen, B.R.D.
- F.R.A. Hopgood, U.K.

The reader should be aware of the fact that these comments had to be provided in a very short period of time. They nevertheless enabled the editorial board to completely compare and comment on the documents concerned in a 3-day meeting.

Krammer et al.

The design of the GSS80 graphic subsystem is a result of experience collected home and abroad. Much use of the Seillac methodology papers, the CORE and the GKS standard specifications has been made. Since this document follows in time those mentioned, it is proper to refer to them as many proper decisions are taken over, or result from their analysis, while certainly our mistakes may not be theirs and if theirs, may be corrected in subsequent editions.

(For information only: a sort of acknowledgement we intend to put in front of GSS80).

THE FORTRAN-SYNDROMA

The standard should be LANGUAGE-INDEPENDENT and reflect STATE-OF-THE-ART or ACCEPTED-PRACTICE. The last two are in contradiction with each other (and we confess: no decision yet).
Both CORE and GKS use FUNCTIONS to describe the standard, and this FUNCTIONS may be mapped one-to-one onto FORTRAN subprograms (with the modification: STRING LENGTH+VECTOR).
Others - especially ILP - use high level constructs.
If the second practice is accepted, it may still allow the standard for the application interface "to be transported..." from the KERNEL to the LANGUAGE DEPENDENT LAYER "... with a small amount of routine alterations ...". (See CORE 1/1.3).

This would not effect the portability of FORTRAN programs between two FORTRAN-machines (machines with FORTRAN compilers). If accepted, however, the whole standard should be reformulated (e.g. a

```
DISPLAY-ON    workstation-identifier
      WITH    viewing-transformation
       THE    picture-type-expression
END-DISPLAY;
```

is not easily explained with FUNCTIONS and TABLES.
Certainly a FORTRAN-like standard is easy to implement in higher level languages.

H.15.01.79. LEVELS OF STANDARD

It is proposed to postpone decision on the "levels of standard".

It is also proposed to form "functional capability groups" (fcg's) within the standard being defined and later define "workstation classes", "levels of standard", etc.

In general, we agree with the GKS practice to omit the discussion of certain questions from the standard's text. Thus: reduce Part II of the CORE volume and augment Part III (as explanatory document).

N°	CORE	GKS	GSS80	remarks
1.	-	-	-	MODEL TRANSFORMS
2.	——— WORLD-COORDINATES ———			
3.	LEFT/RIGHT	RIGHT	LEFT/RIGHT	COORDINATE SYS
4.	——NORMALISED-DEVICE-COORDINATES——			
5.	——————2D-WINDOW——			
6.	——————2D-VIEWPORT——			
7.	——————CLIPP-ON/OFF——			
8.	CLIPP-TO VIEWPORT	CLIPP-TO C.RECTANGLE	?	
9.	NDC 2 specify	NDC 2 retrieve	?	
10.	-	SPECIFY-DISPLAY-SPACE		for next drawing
11.	-	via inquieries	via "metric viewport"	specify size
12.	——————2D-IMAGE-TRANSFORMATIONS——			
13.	in NDC	in WORLD	?	LOCATOR data
14.	VIEW-UP-2D	-	-	

Notes: 1. NDC is conceptual only except for 13 in CORE.
2. Image transformations only at HIGHER LEVELS.

(GKS is left out from this comparison since it does not deal with 3D. This comparison contains some references to PRIMITIVES and INPUT and may differ from the details expressed there. Consider principles only here.)

1./ CORE: 2D and 3D intermixed: once 3D has been used 2D primitives are shorthands for 3D's with $Z = Z_{CP}$.

GSS8O: 2D and 3D completely separated: internal TABLES store both 2D and 3D viewing parameters separately. (Consider LOCATOR to work in a third, "gniweiv" system based on "normalised locator coordinates" and "input window".)
The nD primitives use nD viewing and the redefinition of nD viewing parameters does not change mD (m = 5-n) viewing.

N°	CORE	GSS8O	remark
2./	?	VIEWPORT	on 2D screen
	NDC 3D	?	?
3./	VIEW-REF-POINT	through variables	projection
	VIEW-PLANE	- " -	parameters
	VIEW-UP	- " -	
	PERSP/PARLEL	- " -	
		PROJECT ('PARL' / 'CENT', dir / cent, refp, ...)	
4./	PLANE-to-SCREEN mapping		
	VIEW-PLANE	Y	
	WINDOW	Y	

5./ CLIPPING

ON/OFF	Y
FRONT/BACK PLANE	Y
DEPTH CLIPP ON/OFF	Y

6./ Activate projection

at CREATE SEGMENT	at PROJECT

7./ PRIMITIVES

2D primitives	-
-	MARKER-3D
	LEGEND-3D

Notes: 1. For MARKER 3D(n) and LEGEND 3D(string), the current 3D position is projected and mapped onto the viewport and there a "flat" symbol is drawn; the 2D CP is not effected. Real 3D markers may be implemented later. Texts as parts of the picture may be properly transformed if defined through LINE/MOVE/POLYLINE primitives (from MACRO's?).

2. The activation (and check) of PROJECTION is in effect the same for the both systems.

3. For N° 3-4-5 the visual meaning and easy modification is of prime importance. N° 3 seems good. N° 4 is not too visual, still no better solution.
   An auxilliary VIEW-SPHERE(r) and a VIEW-BRICK (PLFD, PLFU, PRBU)* (in WORLD-COORDINATES) definition of the VIEW-VOLUME is provided but implemented through the truncated piramid concept.

4. Dimetric and other visualization methods (ACCEPTED PRACTICE) provided and implemented through the basic methods already described.

5. In GSS80: VIEWPORT parameters stored at workstation. WINDOW parameters stored at both WORKSTATION and "central-CORE". On ACTIVATE-WORKSTATION effective MAPPING parameters sent to WORK-STATION.
   "List of stored segments" stored at WORKSTATION (as in GKS) as opposed to "list of VIEWING--SURFACES" for each segment (as stored in CORE).

* Point-left-front-down, point-left-front-up and point-right-back-up.

H.15.01.79. GKS FILE

N°	CORE	GKS	GSS80	remark
1.	-	GKS-f	Y/2	see notes

Notes

1./ GKS may use the GKS-file

- to create, store (DELIVER, GENERATE) and reference (INSERT) repetitive pictorial symbols e.g. standard to an application
- to store segments and re-use on the same or on another device
- to remember current status of drawing being edited (example in GKS).

2./ None of these belongs to a lowest LEVEL CORE.

3./ The first (library) function is to be implemented in GSS80 as a higher level function.

4./ The second function needs more than the GKS-file: picture-macros with parameters, conditional expressions etc.
At present programming language procedures are to be used, which is not far from a more idealistic solution: graphics as a proper extension of the programming language.

5./ The third function is exhibited by the example. If this is a hardcopy device, than better have a "hardcopy-attribute" for workstations to produce a snapshot of the screen. The workstation may emulate this via a sort of GKS-file.

If it is documentation drawing than the proper way is to

- edit a modell of which
- only parts are shown on the screen and with possibly different notations (titles, symbolic elements etc.)
- and at the end a different branch of the program is to produce the documentation drawing

Functions

N°	CORE	GKS	GSS80	remark
1.	CREATE SEGMENT	OPEN SEGMENT	BEGIN $..	
2.	CLOSE SEGMENT	CLOSE SEGMENT	END $...	
3.	DELETE SEGMENT	DELETE SEGMENT	Y	
4.	DELETE ALL SEGMENTS	_	?	a
5.	RENAME SEGMENT	-	?	
6.	BATCH-OF-SEGMENTS	-	?	

a./ The REQUEST NEW DISPLAY SPACE ~~display space~~ function of GKS deletes all work station segments.

b./ The functions BATCH, NEW-FRAME, RENAME, DELETE-ALL, etc. resp. UPDATE, REQUEST-NEW-DISPLAY, etc. still not decided for GSS80.

Dynamic segment attributes

1.	VISIBILITY	VISIBILITY	Y	
2.	HIGHLIGHTING	HIGHLIGHTING	Y	
3.	DETECTABILITY	DETECTABILITY	Y	
4.	IMAGE TRANSLATE 2	-	Y	
5.	IMAGE TRANSLATE 3	-	-	
6.	IMAGE ROTATE 2	-	Y	
7.	IMAGE ROTATE 3	-	-	
8.	IMAGE SCALE 2	-	Y	
9.	IMAGE SCALE 3	-	-	
10.	-	TRANSFORM SEGMENT	-	a

a./ Not applicable while segment open.

b./ Inquiery for attributes in GKS is left to the LANGUAGE-DEPENT-LAYER, while in CORE it is part of the standard.

Static segment attributes

N°	CORE	GKS	GSS80	remark
1.	RETAINED/NON RETAINED	STORED/NON STORED	Y	a
2.	IMAGE TRAFO type	-	Y	b,c

a./ In CORE: static attribute, in GKS: function parameter.

b./ In CORE for a segment the type of applicable IMAGE TRAFO must be pre-declared.

c./ In CORE IMAGE TRAFO affects the clipped image. GKS does not explicit about that. In GKS all STORED WORK STATION SEGMENTS may be TRANSFORM-ed (if WORK STATION is capable).

Two different typings of GKS differ in that TRANSFORM affects GKS FILE SEGMENTS or not. If GKS FILE SEGMENTS may be TRANSFORM-ed, than this happens definitely before clipping.

Main differences

N°	CORE	GKS	GSS80	remarks
1.	3D including 2D	2D only	3D and 2D separated	a
2.	CURRENT POSITION	_	Yes	b
3.	form CP to endpoint	-	Y	"relative", b
4.	-	form X_1 to X_2	-	b
5.	from CP with ΔX	-	?	"incremental", b
6.	-	GKS file segment	Yes	c

a./ Separation of 2D and 3D increases CLARITY of the STANDARD and EFFECTIVENESS of an implementation.

b./ Both CORE and GKS has a COMPACT and COMPLETE set of output primitives.

c./ In CORE explicitly exluded (1.4.2.).

Output primitives

N°	CORE	GKS	GSS80	remark
1.	MOVE ABS 2	-	MOVA2	
2.	MOVE ABS 3	-	MOVA3	
3.	MOVE REL 2	-	?	a
4.	MOVE REL 3	-	?	a
5.	LINE ABS 2	-	LINA2	
6.	LINE ABS 3	-	LINA3	
7.	LINE REL 2	-	?	a
8.	LINE REL 3	-	?	
9.	POLYLINE ABS 2	POLYLINE	PØLA2	b
10.	POLYLINE ABS 3	-	PØLA3	
11.	POLYLINE REL 2	-	?	a
12.	POLYLINE REL 3	-	?	
13.	TEXT	STRING	STRING	
14.	-	-	MARKER	c
15.	MARKER ABS 2	-	-	
16.	MARKER ABS 3	-	-	
17.	MARKER REL 2	-	-	
18.	MARKER REL 3	-	-	
19.	-	POLYMARKER	-	
20.	-	INSERT SEGMENT	Y	d
21.	-	DRAW	CURVE	e

a./ The necessity of relative coordinates is deleted yet.

b./ CORE and GKS POLYLINE differs in the starting point.

c./ MARKER in GSS has only one parameter: the marker number.

d./ A macro facility is an important part of a RICH and COMPACT system. We want to implement such an extension, but the form and level is not yet decided.

e./ A curve generator function will be implemented in GSS80.

Static primitive attributes

N^{o}	CORE	GKS	GSS80	remark
1.	CURRENT PICK ID 1.5 level	- 1 level	Y 1.5 level	a PICTURE STRUCTURE
2.	CURRENT COLOUR	-		b
3.	CURRENT INTENSITY	-		
4.	CURRENT LINESTYLE	-		
5.	CURRENT LINEWIDTH	-		
6.	-	PEN REPRESENTATION		b, c
7.	-	PEN NUMBER		
8.	CURRENT CHARPLANE	-		
9.	CURRENT CHARSPACE	CHARACTER SPACING		
10.	CURRENT CHARSIZE	CHARACTER SIZE		
11.	CURRENT FONT	TEXT FONT		
12.	CURRENT CHARQUALITY	TEXT QUALITY		

a./ If number of candidate units for manipulation is in the range of number of candidate units for operator-pick, the two systems are equi-potent. If, however, the second is relatively high, 1.5 levels may save plenty of storage and some time aswell.

b./ In GKS it is easyer to change program for different devices.

c./ GKS text doesn't specify clearly whether the linetype(style) and linewidth affects texts or not.

1./ Input mode

GKS: "single wait mode", that is: one device ENABLED at a time, and program suspended until the end of input action.

CORE: "non-wait mode": any number of devices ENABLED at a time, program runs until AWAIT encountered. Input data queued.

GSS80: "selective wait mode": any number of devices ENABLED, program runs until AWAIT encountered, however the first input action DISABLES all devices. No input queue, the DEVICE NUMBER received by AWAIT is used to retrieve data from the proper device.

To avoid loss of data it is advisable to AWAIT immediately after the required number of ENABLE's and retrieve data immediately after AWAIT. (Not a beautiful solution.)

2./ Logical input device classes

N°	CORE	GKS	GSS80	remark
1.	PICK	PICK	PICK	b
2.	KEYBOARD	KEYBOARD	KEYBOARD	b
3.	BUTTON	CHOICE	BUTTON	a,b
4.	LOCATOR	LOCATOR	LOCATOR	b
5.	VALUATOR	VALUATOR	VALUATOR	b
6.	-	-	CLOCK	?

a./ BUTTON in CORE is not ORTHOGONAL in the sense that the knobs on a box are different DEVICES, not different codes from one device.

b./ In GKS all input devices used in a POLY-input mode.

3./ Device characteristics and special functions

Prompting, initial value setting, echoing, inquiry are highly implementation dependent features
GKS is more honest to confess dependency, CORE functions are very much implementation dependent.

4./ Input functions

N°	CORE	GKS	GSS80	remark
1.	device class	-	-	
2.	ENABLE	-	Yes	
3.	DISABLE	-	-	
4.	AWAIT	-	YES	a
5.	queue	-	-	
6.	ASSOCIATE	-	-	
7.	DISASSOCIATE	-	-	
8.	single input data	-	Yes	
9.	—	input group of data	-	
10.	READ LOCATOR	-	Yes	
11.	READ VALUATOR	-	Yes	

N°	CORE	GKS	GSS80	remark
12.	GET PICK DATA	REQUEST SET OF PICK DATA	GET PICK DATA	see 8., 9.
13.	GET KEYBOARD	REQUEST TEXT	GET KEYBOARD	see 8., 9.
14.	-	REQUEST SET OF CHOICE	GET BUTTON	see 8., 9. b
15.	GET LOCATOR	REQUEST SET OF LOCATOR DATA	-	
16.	GET VALUATOR	REQUEST SET OF VALUATOR DATA	-	

a./ GSS80 doesn't have TIME parameters, the CLOCK device may be ENABLED instead.

b./ CORE data received by AWAIT.

5./ Minimum set of input devices

N°	CORE	GKS	GSS80	remark
1.	1	1	1	PICK
2.	1	1	1	LOCATOR
3.	1	1	1	KEYBOARD
	?	ASCII 96	ASCII 96	"resolution"
4.	8	1	1	BUTTON
	1	?	8	"resolution"
5.	4	1	1 (?)	VALUATOR
	6 bits	?	6 bits	"resolution"

GKS/CORE COMPARISON - SEGMENTATION

D. S. H. ROSENTHAL

INTRODUCTION

The following is an attempt to identify the semantic differences between the graphics pseudo-packages specified in the ACM-SIGGRAPH Core Report and the DIN Proposal in the area of segmentation. It consists of a description of this area of each package in a compatible format, and a comparison of the two.

GKS SEGMENTATION

In GKS all segments have names, and the segment associated with a particular SEGMENT_NAME can be in one of five distinct states:

1. Non-existent.
2. Open.
3. "On" at least one work station.
4. "In" the GKS file.
5. "On" a work station and "In" the GKS file.

The GKS description is not completely clear about multiple work stations; the following assumes only a single work station.

Transitions between these states are caused by applications of the following GKS functions:

A. OPEN_SEGMENT
B. CLOSE_SEGMENT
C. DELETE_SEGMENT
D. RESET_GKS_FILE
E. CLOSE_WORK_STATION
F. GENERATE_GKS_FILE_SEGMENT_FROM_RECORD
G. REQUEST_NEW_DISPLAY_SPACE

The transitions between states caused by each function are set out in the following table:

			TO STATE				
			1	2	3	4	5
	S	1	-	A	-	F	F*
F	T	2	-	-	B	B	B
R	A	3	CEG	-	-	-	-
O	T	4	CD	-	-	-	-
M	E	5	C	-	D	EG	-

not true * I assume that function GENERATE_GKS_FILE_SEGMENT_FROM_RECORD in state "Work Station + GKS File Active" results in a work station segment as well as a GKS file segment, the document is not clear on this point.

CORE SEGMENTATION

In the Core, segments are of two main types. Non-retained segments have no names, but the retained segment associated with a particular SEGMENT_NAME can be in one of three distinct states:

1. Non-existent
2. Open
3. Closed

Transitions between these states are caused by application of the following core functions:

A. CREATE_SEGMENT
B. CLOSE_SEGMENT
C. DELETE_SEGMENT
D. DELETE_ALL_SEGMENTS
E. RENAME_SEGMENT

The transitions which they cause are set out in the following table:

		TO STATE		
		1	2	3
FROM STATE	1	-	A	-
	2	CDE+	-	B
	3	CDE	-	-

+ See resolution of issue 3.1.6.

COMPARISON

1. The Core permits access to the open segment by name (DELETE_SEGMENT, RENAME_SEGMENT and inquiry). GKS only accesses the open segment implicitly, and permits DELETE_SEGMENT only when no open segment exists.

2. GKS provides no RENAME_SEGMENT function, whereas the Core does.

3. The Core distinguishes two types of segment, "retained" and "non-retained". GKS distinguishes two types of segment, "work station" and GKS file", though a single name may have both types of segment simultaneously associated with it. These concepts differ as follows:

   3.1 All GKS segments have names. Only retained Core segments have names.

   3.2 GKS work station segments survive until explicitly deleted or their work station is closed. Non-retained Core segments disappear at a new frame action.

   3.3 GKS work station segments may undergo image and visibility transformations, if the work station supports them. Non-retained Core segments may not undergo such transformations.

   Thus it appears that all GKS segments are "retained".

4. The support of image transformations is a property of the GKS work station, and all segments on that work station may be transformed. The support of image transformations is a property of the Core level, and only segments of appropriate type may undergo image transformations.

5. The Core provides, from level 2 up, only one place where segments may be stored, in the display file. GKS provides three places, namely "On" the work station, "In" the GKS file, and in a user record. GKS rejects the argument (a) of Core issue 3.2 and insists that all implementations include at least one display file (the GKS file). This decision means that there is no prohibition on functions parsing the display file (e.g. INSERT_SEGMENT).

R. Williams

SIGGRAPH

Comments on the Graphical Kernel System (GKS) and the GSPC proposal with respect to their differences.

NOTE: These reactions are mine only reporting as an individual and do not represent a view of IBM.

Output primitives and Generation Functions

GKS has no "current position" concept and no MOVE function; GSPC has both. While GKS is cleaner (less confusion); GSPC tries to encompass current practice.

Input Primitives

Is GKS CHOICE input a kind of "button" or switch input. It needs clarifying.

Input functions

In GKS input occurs when a device is used; in GSPC input devices are enabled and then separately read. GSPC also has "event handling" and an "Await-event function".

Storage of Graphical Information

GKS has two levels of storage "segments" on the work station and "segments" in the GKS file. Also mentioned is record I/O in GKS file functions, but "record" and format/structure of a record is not discussed. GSPC has segments only in one sense (the GKS file sense). GSPC has no work station but leaves the 'display file' management to the device driver level. GKS assumes a memory in the workstation but a storage tube terminal and a plotter may not have a memory. However the two proposals are not very different but only seem to be. For instance in GKS one cannot read back a segment from the work-station memory, it is for output only and therefore a GKS station is like a GSPC device view-surface. Also GSPC has the concept of retained segments or non-retained segments (analogous to GKS states: work station active or not with GKS file active). In GKS segments are named in GSPC segments and primitives in a segment are named and identified respectively for more refined 'picking' in interactive applications however GKS "Draw" has an unexplained "generalized primitive identifier". A Pseudo-segment file in GSPC (I'm not certain of its latest characteristics) allows segments to be multiply reusable, as also are GKS file segments. The files, view-surface, work-station could benefit by a common nomenclature and definition system. The intentions appear

to be close enough to become one. Where do intermediate files for device drivers fit with respect to a work-station?

Transformations

In GKS the order of transforming a segment (object) is scale, rotate and shift. In GSPC it is rotate, scale and shift. These two specifications are equivalent but could be more consistently stated for purity.

GSPC uses graphics to explain the viewing transforms. (Very helpful. When GKS is expanded to 3D it should be consistent with the GSPC transformation.

New Display Space

I like the word ERASE or even CLEAR better. GSPC uses new-frame. "Clear" is acceptabe for plotters too. GKS "update" is the same as GSPC new-frame.

Inquiry

GKS states that this is to be defined in the language dependent layer; however state information is accessible. Terminal characteristics should be made available. (# characters in row, # rows etc.) GSPC includes a wide variety of inquiry functions. A consensus needs to be reached because the approaches are quite different.

Non-Standard Issues

GSPC provides a standard method to be non-standard - i.e. an escape function. GKS does not provide such a mechanism.

Modelling Transformation

A "hook" is provided by GSPC but not by GKS. Composite modelling/viewing transforms are possible.

Overall Impressions

GKS gives the appearance of being clean and simple and adequate for 2D interactive graphics. It gives one the impression of being easier to learn initially than GSPC. The GKS authors are to be congratulated on another fine effort. It is hoped that we all benefit from the best of both GKS and GSPC (and other porposals).

Robin Williams

Comparison of GSPC Core and DIN GKS

K.W. Brodlie (Leicester University)

1. Introduction

Both standard proposals specify a set of basic functions for a graphics system, independently of graphical output device, programming language or application.

Main points are:

(a) Both proposals group output primitives into segments, but only GKS allows the user to keep a file of segments and hence insert previously created segments into new segments.

(b) GKS is restricted to 2D; the Core allows 3D.

(c) Both systems use a window-viewport type of mapping from user co-ordinates to normalised device co-ordinates.

(d) The Core has a CP; GKS has not.

Some particular areas are compared in more detail in the following sections.

2. Initialization of system, work stations, view surfaces.

Both the Core and GKS have similar routines to initialize and terminate the graphics system. The Core is structured into levels and so the user has to specify via the initialization routine the level of the system he is using. GKS allows a user-defined error procedure to be specified at initialization.

The Core supports just one work station, but that work station may have several view surfaces, any number of which may be selected at any one time. GKS supports a number of work stations each with one view surface, but only one work station can be active at any time.

3. Display space control

In GKS, 'SET REQUESTED DISPLAY SPACE' lets the user define a subrectangle, relative to the actual display space, for each work station. A paper quality can also be defined for plotters. The Core has 'NDC SPACE 2' which specifies a maximum width and height in normalised device co-ordinates and which applies to all view surfaces. The GKS scheme seems more flexible, but it is not clear if the normalised device co-ordinate space on a work station is in terms of the actual or requested display space - presumably it is the requested space. Also it is not clear if the normalisation is such that the maximum or minimum dimension is 1.0.

4. Viewing transformations

Both systems have a window-viewport mapping.

The Core allows the user-space to be rotated with respect to device-space (via the VIEW UP 2 vector) before the window is applied. There is no corresponding function in GKS.

Both allow clipping, but in the Core there is simply an on-off switch for window clipping. In GKS, the user can specify a clipping rectangle within the window. This is a big improvement because it separates the notions of mapping and clipping.

5. Segments

GKS allows the user to maintain a display file, the so-called GKS file. Segments written to this file can be recalled either for insertion in another segment, or for 'delivery' to the application program, presumably for long-term storage. Likewise a segment can be passed from the application program to the GKS file.

As far as can be seen, inserted segments lose their own identity and assume the attributes of the segment into which they are inserted. Thus the GKS INSERT function is simply a shorthand for a sequence of output primitives. This needs to be made clearer in the definition of GKS.

The Core does not maintain a display file, and thus the insertion of segments is not a possibility.

Both systems have a function to delete a segment, but only the Core has a function to rename a segment.

Both systems have a function to transform segments at a work station: TRANSFORM SEGMENT in GKS, and IMAGE TRANSFORMATION in the Core.

The Core allows primitives within a segment to be detected using the PICK ID attribute. GKS has no such function.

Segment attributes of visibility, detectability and highlighting seem similar in the two systems.

6. Primitive attributes

In the Core, all attribute setting is done within segments. In GKS, colour, intensity, linewidth and linestyle cannot be specified directly within a segment. Instead the user must define a set of conceptual pens outside a segment, these pens having attributes of colour, intensity, etc., and any change within a segment is achieved by pen selection. This is a good scheme because it allows segments to be defined in a device-independent manner.

In both systems, character attributes are specified within segments. The attributes are similar - size, spacing, font and quality. One small difference is that in GKS high quality text is subject to the current line attributes (i.e. linestyle, linewidth) - not so in the Core. (Incidentally it could be argued that text font and quality should also be associated with a pen and specified outside a segment).

7. Output primitives

GKS has no CP, while the Core has one CP. As a result the GKS POLYLINE primitive replaces six Core functions MOVE, LINE, POLYLINE in absolute and relative co-ordinates.

The definition of a marker is similar in both systems - i.e. device dependent and without size or quality. In my view this is wrong - markers should have size and quality attributes.

GKS has a DRAW function to act as a generalised output primitive. There is no corresponding function in the Core. Probably a function such as DRAW is best specified in a higher level standard.

Extract from Minutes of 5th Meeting of
BSI DPS13/WG5, held on 22nd January 1979

Minute 4: Comparison of the DIN GKS and GSPC Core Standard Proposals.

The meeting discussed the new German standard proposal GKS, and compared it with the GSPC Core. The following summary of the BSI group's joint views will be sent to Paul ten Hagen as co-chairman of the ISO Editorial Board, together with personal contributions from Bob Hopgood, David Rosenthal, Ken Brodlie and perhaps others. Hopgood's paper was distributed at the meeting.

It was agreed that the major difference between the two proposals was the decision by GKS to maintain a display file, the so-called 'GKS file'. This almost certainly influenced many of the other decisions taken in GKS, and many of the differences between the two proposals can be traced back to this single major distinction. The idea of maintaining a display file was generally welcomed.

The following other points were agreed:

(a) Current Position: The GKS treatment of CP (i.e. no CP) is tidier and more consistent.

(b) Pen Representation: The GKS idea of setting pen attributes outside segments is a natural consequence of the decision to maintain a display file, and in this context is welcomed. However further explanation of the advantages of this form of attribute setting in the case of raster graphics is needed, and in particular it is unclear if a change of pen representation affects closed segments. The pen representation style of attribute setting is useful for writing device independent graphics subroutines.

(c) DRAW: The generalized output primitive DRAW in GKS should not be part of a standard at this level. Its correct place is at a higher level standard, although there must be a 'hook' to allow a by-pass of the lower level. (This is in order to make use of circle-drawing, ellipse-drawing, etc. by intelligent devices).

(d) Renaming: The lack of a RENAME function in GKS is considered a serious omission. Such a function is particularly useful for extending segments which are already closed, something which cannot adequately be done using the INSERT function.

(e) INSERT: The INSERT function in GKS is another consequence of the display file decision. It is assumed that the INSERT function is simply a 'macro'-type facility for inserting output primitives into a segment, and does not offer true nesting of segments - but this needs to be made clearer. The INSERT function means that symbol drawing is made very much easier in GKS than in the Core.

(f) 2D versus 3D?: It is correct to establish a standard for 2D graphics before tackling 3D.

(g) Levels or Modules?: Clarification is needed from the Germans on further development of GKS - whether the structure will be in the form of levels or modules. The minimum needed for an implementation of GKS appears to be greater than for level 1 of the Core.

(h) Input: A PICK ID function is needed in GKS to make the handling of menus easier.

There should be some facility in GKS for defining graphical responses for prompting, and associating these with various input devices.

It is a poor feature of GKS that only one input primitive can be enabled at any one time. It should be possible to enable a set of primitives, and get GKS to interpret which primitive is received.

(i) Error-handling scheme: The option of a user-defined error procedure in GKS is welcomed.

There were mixed views on the merits of a separate clipping and window rectangle as in GKS, with a small majority in favour.

Encarnação Wisskirchen

Comparison of properties between the GKS and the GSPC proposals

I 3D

1) GKS is a pure 2D-system and the basis for a standard. 3D is not yet included and not to be handled in the near future.
Motivation: Most criterion

2) The GKS user interface does not know about a CURRENT POSITION and has no commands for MOVTO and LINTO and no relative coordinates.
Motivation: CURRENT POSITION, MOVTO, LINTO ...
have a strong device dependent component. (device simulation)

3) The GKS user interface has an easy understandable input interface (request input). The "state of the art", e.g. the event input has a strong device dependent component.
Motivation: The aim is to have a symmetry between input and output commands, e.g. additional effort is needed to make a standardization of high-level input primitives possible.

image Entity II

4) The GKS user interface includes an object storage (GKS-File Object library). Objects may be used for the (primitive) modelling of other objects.
Motivation: Most criterion

5) The GKS system includes the concept of a workstation (formed by any set of output and input devices). Fundamental capabilities:
   a) ability to receive a stream of standard output (picture) data
   b) ability to send a stream of standard input data to another modul
   c) ability to store data

I query Entity

6) The GKS system has only the one level naming and modification mechanism
Motivation: Symmetry

7) The GKS system has no HOOK, e.g. there is no standard mechanism for the integration in GKS of a "non standard capability" of a graphic system. High-level graphic output primitives may be implemented by the DRAW commands (generalized primitives).

8) The GKS system is characterized by states and operations. A special importance is given to the operating states

- GKS closed
- GKS open
- workstation active
- GKS file active
- Workstation and GKS file active
- Segment open with Workstation active
- Segment open with GKS file active
- Segment open with Workstation and GKS file active

The GKS system has so a strong syntactical structuring.
Motivation: Least astonishment

9) The GKS system allows only to activate one workstation at a time.

SCIENCE RESEARCH COUNCIL

COMPARISON OF DIN AND GSPC
(BSI Meeting Jan 22 1979)

January 3, 1979

F R A Hopgood

1 INTRODUCTION

This paper does not attempt to give a full comparison of the two systems. One reason is that the degree of completeness in terms of the definition differs considerably. For this reason, precise comments can often be made about one system and not the other. Also, it is possible to interpret both systems in a variety of ways. A larger set of examples showing implementation strategies on a variety of devices (particularly storage tunes) would be welcome.

2 LEVELS

Both systems are defined on a level rather than module basis. Consequently, a storage tube or plotter user has to carry the full system. GSPC have shown signs of going away from the level approach. In the DIN proposal, there is no method of using it without implementing segments in some way. The DIN proposal is approximately levels 1 to 3 of the GSPC proposal but omitting 3D. The lack of any subdivision of the DIN proposal may be a criticism.

3 OUTPUT PRIMITIVES

The DIN proposal has made the major change of defining no current position and insisting that all primitives are defined in absolute coordinates in the user coordinate system. The problems experienced with current position indicate that this is a sensible way to go and is preferable to the GSPC approach.

As a result of this decision, many of the GSPC primitives translate to a single DIN primitive. For this reason, there is little difference between the proposed primitives in the two systems. Both include polyline. The DIN proposal does include a generalised primitive identifier which allows a whole series of higher level primitives. It is interesting to note that these include area fill as well as more complex arc drawing primitives. It is important that these additional primitives are well defined. No information is given concerning these and it is suggested that they may be part of a separate standard. Would there be one or possibly more than one standard for these higher level primitives. Are they seen as derivatives from polyline or separate primitives with their own attributes. Is it possible to define new attributes for them.

The DIN proposal does not allow the concatenation of text and would expect this to be done by buffering prior to output if even spacing of low quality text is required.

4 PRIMITIVE ATTRIBUTES

Both systems define colour, intensity, line type, thickness, font, char size as primitive attributes of some kind. The DIN proposal appears to define all but character attributes as segment attributes. If this is true, it is likely to force the proliferation of segments with the corresponding overheads. The DIN proposal has a pen number attribute which implies an open-ended set of device dependent options. There is insufficient detail to make any comment.

Major difference between the two proposals is the lack of a PICK-ID in the DIN proposal. This means that each light button of a menu will reside in separate segments with the consequent overheads which, in the DIN proposal, appear large. This is a fundamental omission which needs to be assessed carefully. Most existing systems which lack this have found a need for it.

5 SEGMENTATION

The major difference between the two systems is the introduction of a central file of segments in the DIN proposal. It allows segments to be manipulated and output to a number of devices using this facility and the INSERT function.

The DIN proposal specifically allows the user to control the flow of traffic from system to device. Renaming and appending to segments is presumably done by using the INSERT facility in the DIN proposal.

Both proposals provide image transformations of individual segments. This goes against the concept of divorcing picture drawing from modelling. For example, see the DIN test program in their document.

6 VIEWING TRANSFORMATIONS

The DIN proposal is only 2D and consequently much simpler. ThThere may be problems in extending to 3D if compatibility with 2D is to be retained. The DIN proposal has a separate clipping window from the viewing window.

7 INPUT PRIMITIVES

Both systems have the same 5 input functions. The DIN proposal does not have the concept of an event and a sampled device. Instead a request is made for a specific number of inputs from a particular device and only one device can be active at a time. This seems to be far too restrictive and will certainly cause problems. It severely restricts the operator's method of working.

Valuators in the DIN proposal are restricted to the range (0,1) for no apparent reason.

There is no minimal set of available input devices in the DIN system. Consequently, many device dependent features will have to be implemented in the application program. The GSPC approach of specifying a minimal set always supported seems preferable.

The GSPC proposal allows some connection between input and echoing. Most of the DIN proposals see this as device dependent. Consequently, the operator may get a significantly different view of the system on different devices.