TECHNICAL MEMORANDUM

27

An evaluation of the GKS proposal for standards in graphics with respect to the GSPC core proposal

A. Lemaire, H. Watkins,
A. Ducrot and E. Saltel

Operations Department

April 1981

European Centre for Medium-Range Weather Forecasts
Europäisches Zentrum für mittelfristige Wettervorhersage
Centre européen pour les prévisions météorologiques à moyen

AN EVALUATION OF THE GKS PROPOSAL FOR STANDARDS IN GRAPHICS WITH RESPECT TO THE GSPC CORE PROPOSAL

BY A. LEMAIRE & H. WATKINS (ECMWF)
AND A. DUCROT & E. SALTEL (INRIA)

INTRODUCTION

In order to implement a basic 2-D device independent graphics package at the European Centre for Medium Range Weather Forecasts (ECMWF), an evaluation was made of the West German GKS proposal for standards in graphics /1/ with respect to the ACM-SIGGRAPH GSPC CORE proposal./2/ /3/.

Both proposals have been based on the concepts and methodological principles which have emerged from the IFIP SEILLAC I WORKSHOP on methodology for graphics /4/. Yet they differ in many aspects and are both subject to criticisms.

GKS was chosen as a guideline for this evaluation because it is basically a two-dimensional graphic system whereas the CORE proposal is basically 3-D. Although weather forecasting deals essentially with a three dimensional world, the viewing transformations which are used, e.g. polar stereographic projection, cannot be performed by the CORE 3-D viewing system.

In the following we shall go through the GKS proposal version 5.2 (document DIN 00 66 252) extracts given in Annexes 1 and 2. In order to make referencing easier, the section titles and numbering have been kept identical to those of the DIN document. In the case of a reference to the CORE proposal, the version number, i.e. 77 or 79 and the page number will be specified.

1. SCOPE OF APPLICATION AND PURPOSE

Graphic software concerns a wide range of applications and has to interface to a large variety of graphical devices. Any attempt to design standards for graphics should aim to be usable and efficient for "most applications and most devices" as stated by the GSPC.

Let us first consider the application side. Both GKS and the CORE perform two distinct tasks namely, a viewing transformation and, conceptually, the

basic functions of a "model" graphical workstation. It has already been mentioned that the 3D viewing transformations provided by the CORE are not applicable in most meteorological applications. Unfortunately this also applies to the simpler 2D window to viewport transformation provided by GKS and the 2D subset of the CORE. For meteorological applications it is often more efficient and memory saving to perform the clipping operation at the early stage of data generation than when the graphical output is produced. For instance it is preferable to contour only a part of a big array than to contour the whole array and then clip to the restricted area. Also,non rectangular windows are sometimes required. Of course meteorology only represents a limited area of computer graphics applications but the argument surely stands in other cases. The point is that although the basic graphic functions defined in GKS and the CORE are suitable for most applications, the proposed viewing transformations are less widely applicable.

From a methodological point of view the SEILLAC I workshop has recommended the valuable approach of identifying and clearly setting apart the concepts encountered in computer graphics applications. This approach has been adopted by GKS and the CORE. The "modelling" system has been separated from the graphic system. In so doing the "modelling" transformations have been distinguished from the viewing transformations, although they imply the same mathematical operation. However, a major confusion still remains in both proposals with respect to the output primitives and the segment concept. This is due to the fact that no distinction has been made between the viewing system and the basic graphic system. As a result, the output primitives are in fact means of describing objects to the viewing system in a world coordinate system and not means of describing true graphical entities on a view surface. Similarly segments are associated with a snapshot of an object and not with the view surface management.

From the application programmer's point of view it is very useful and natural to directly access the view surface. Recalling the GSPC synthetic camera analogy, why shouldn't it be possible to edit a "synthetic" family album by positioning pictures and adding text and drawings in order to comment or enhance the presentation of the page?

Separating the viewing system from the basic graphic system would result in a much cleaner, less confusing and more flexible graphic system. Access to the view surface is possible in GKS and the CORE by setting a "transparent" viewing system and is indeed the default setting. This solution however is not a clean one with respect to the methodology and it might

result in unnecessary overheads.

If we now turn to the device aspect, one question arises : should the standard consider raster devices? The "device independence" principle certainly implies that raster devices should be considered. However, when it comes to the specification of output primitives one should avoid the introduction of concepts which are either highly specific to one particular type of device or do not fit well inside the global system. In other words, only those primitives which can reasonably, if not efficiently, be represented on any device and have clearly defined relationships with related concepts of the system, e.g. other primitives, coordinate system, etc..., should be introduced. To this respect the GKS "fill area" primitive is acceptable, whereas the "pixel array" is not. This will be discussed in more details in section 3.1.

3. THE GRAPHICAL KERNEL SYSTEM

3.1 Graphical output

The concept of current position(s) has been quite rightly rejected by GKS. As stated in the GSPC report /79,P II-111/ :

"(the use of) no CP is cleaner, more consistent, and less confusing".

Since this matter has already been argued so many times it will not be discussed any further.

In principle, TEXT and DRAW could be the only output primitives which need be defined. The generalised drawing primitive DRAW would then be defined as an ordered set of definition points (geometry) plus extra information (morphology) such as interpretation e.g. polyline, polymarker, spline,... and substance e.g. pen number, as described in /5/. Not only does this approach follow the methodological principle of separating the concepts but it has also the practical advantage of allowing solid areas to be limited by circles, splines, etc... Needless to say the GKS "DRAW" primitive does not correspond to this proposal as it is in fact a "hook" for allowing access to special features of existing devices. This point will be further discussed in section 3.5. See also section 3.4 about the primitive DRAW, the NDC and clipping.

As regards the TEXT primitive, it is clear that this concept is much more significantly and conveniently defined on the view surface than in any user-defined world coordinate system. Texts are normally used to comment, label and communicate with the operator. Whenever a text is really part of an

object, it should loose its identity at the basic graphic level and be described in terms of lines or solid areas to which the viewing transformation applies. To be consistent, GKS and the CORE which present a viewing system interface to the user, have to define the text in the world coordinate system. This problem is easily solved by providing a viewing system function returning the view surface coordinates of a point defined in the world coordinate system.

As stated before, the primitive "PIXEL ARRAY" should not be included in the standard system. The pixel array corresponds to a very specific utilisation of raster devices which violates the "device-independence"principle. Furthermore, GKS lacks the associated functions for making it a usable feature. Even so, due to the lack (or the too large variety) of experience in the area, it is doubtful that any agreement could be reached on these functions at the moment. Finally, the coordinate systems in GKS are certainly unsuitable for pixel arrays.

The GKS pen and text attributes are concepts which somehow correspond to the philosophy of separating geometry and morphology. They allow more device independent programming and flexible adaptation to the capabilities of the device in use.

3.2 Graphical input

Just as a matter of terminology, the word "graphical" shouldn't be associated with input since most input primitives and input handling techniques are not of a graphical nature.

3.2.1. Input classes and types

To be consistent with the corresponding output primitive POLYLINE, the input primitive LOCATOR should return a sequence of positions rather than one single position. Using the input technique "request tuple of locators" to perform this function is not equivalent. Indeed, entering P successive positions each being carefully adjusted and validated is not equivalent to entering a single hand-drawn curve. It should be noted that defining an input primitive polyline (or curve) to replace LOCATOR would still be compatible with the "request tuple of" function and would correspond in this instance to entering a hand-drawn symbol, i.e. a series of input curves.

For the sake of consistency and for practical reasons, an input primitive equivalent to the output primitive DRAW should also be defined. The same arc or spline generator could advantageously be used for echoing while the

definition points are entered and eventually updated.

Unlike the CORE, GKS does not restrict an input device to be either a sample or an event device. Although this results in some rather loosely defined sampling functions such as for CHOICE, PICK or STRING, it is a more general solution which avoids having constantly to refer to the manual in order to verify the class of a given device.

GKS does not provide for associating sampling with events whereas the CORE does. This facility appears essential in the case of a remote workstation. However, when a remote workstation is used the best solution, if not the only satisfactory one, is to implement on the satellite computer a significant part of the application's command language interpreter. From the hardware capability and cost point of view, the present development of microprocessors makes this solution a perfectly reasonable one. With respect to the programming of highly interactive applications, the CORE association feature appears rather misleading.

The question of which coordinate system to use for input will be discussed in section 3.4.

3.3 The segment concept

In relation to the discussion in section 1, the segment concept may correspond either to a snapshot of an object or to the grouping of graphical output primitives which could eventually originate from various sources, i.e. viewing systems.

In the case of a snapshot, the following rules should apply :

i) No output primitives outside segments, i.e. an image cannot be produced without a camera;

ii) The viewing transformation cannot be changed within a segment, i.e. the camera must not be moved while taking a snapshot;

iii) A segment cannot be appended, i.e. after a snapshot an advance film takes place.

In the case of grouping output primitives for an easier and more efficient way of updating the image, the previous restrictions are irrelevant. GKS has choosen this latter definition while the GSPC has opted for the former. This

confusion in defining the segment concept is a direct consequence of the confusion between the viewing system and the basic graphic system. In fact both definitions are acceptable, very useful and compatible. Provided that they are cleanly and separately defined, the snapshot concept and the segment concept could indeed coexist in a combined viewing and basic graphic system.

The segment concept defined as the grouping of output primitives is a set described by extension, i.e. through the enumeration of its components. There is conceptually no objection to allow the extend (or append) function. Indeed, since the create-close functions are just syntactical short-cuts, the segment name could be associated with each primitive as another attribute. This property does not apply to the snapshot concept.

The INSERT function, which is clearly a modelling function, should be removed from GKS. Surely this function is a very useful feature which should be offered to the programmer one way or another but, in no case, through the graphic system. An explicit and dedicated segment storage workstation could advantageously be used for performing the insert function outside the graphic system.

As a result, the segment storage should also be removed from GKS since the INSERT function is its only justification. The fact that a workstation implementation makes use of an internal segment storage for performing the transform function is irrelevant to the programmer. The fact that the same software is used for implementing the segment storage on many workstations is also irrelevant but should be recommended as good implementation practice.

3.4 Coordinate systems and transformations

Both GKS and the CORE define a three level coordinate system, namely :

i) One world coordinate system (WC)

ii) One normalized device coordinate system (NDC)

iii) One device (or screen) coordinate system (DC)

However, the CORE is more restrictive in that it does not provide a workstation transformation. The problem when defining which coordinate system to be used on the viewing surface is that it is an application dependent choice. In some cases, mainly when screens are used, the actual size of the

picture is irrelevant and an NDC system is perfectly adequate. In other cases, mainly when plotters are used, the size is very important and what's more the usage of an NDC is confusing even if the NDC can be set equivalent to a display area defined in metric units. However, this is more a question of convention than a fundamental issue and there is no reason why a more flexible and useful solution couldn't be adopted. Let the user define his own virtual display surface coordinate system and let the default units be metric thus allowing the user to think in terms of a natural and convenient medium for the output, namely a sheet of paper. Later on, the user controlled workstation transformation could eventually be used for adapting to the actual device display surface. Incidently, in the case of a basic graphic system as proposed in section 1, some primitive attributes could then be defined in metric units, which undoubtedly is far more convenient.

With respect to the DRAW primitive, it should be noted that, as its definition points are not and must not be clipped, this might result in coordinate values outside the NDC range. Furthermore, in order to properly clip arcs and smoothed curves, the clipping rectangle associated with the viewing transformation must be passed to the workstation.

With respect to input, a distinct input surface should be defined. First the separation between input and output follows the basic methodological principle. Secondly, from the programmer's point of view it would be more convenient. It is true that, sometimes, the input is related to the actual picture, but generally the input surface, say a tablet, is given a layout which is different from that of the output surface. Furthermore, returning input coordinates in the world coordinate system is very often meaningless due to this discrepancy between the output and the input layout but also because more than one window to viewport transformation might have been used for describing the picture and thus no one could be selected a priori. Again, not separating the viewing system from the basic graphic system causes problem. What is required is a separate input surface. As for the display surface, the user could define his own coordinate system. The graphic system should also supply an inquiry function returning the actual size of the input device and a function converting coordinates from the input surface to the world coordinate system (viewing system function).

3.5 The workstation concept

Compared to the CORE where it is an implicit concept, the GKS explicit workstation concept is more meaningful and bears more capabilities, e.g. workstation transformation, pen and text representation, etc... considering

the present evolution of microprocessors and the fact that graphical devices can and are now associated with more and more computing and storage facilities, this is obviously a move in the right direction. Indeed, the workstation is the most convenient and efficient place to perform such functions as line drawings with various linestyles, generation of arcs, smoothed curves and low to high quality text, clipping, interaction, etc... However, as the level of these local functions increases, they become more and more device and application dependent. If they have to be accessed through the unique channel of the graphic system, there is a danger that this system will either contain a lot of "hooks" or become itself device and application dependent. The high quality text and the linestyle specification in the core are good examples of what could happen. There is, however, an alternative approach to the problem which, to some extent is the one adopted by GKS. Observing that most data to be passed to the workstation for performing its high level functions are parameters e.g. attribute values, rather than function calls, it is possible to admit that these parameters be passed to the workstation outside the graphic system channel and even be specified directly at the workstation level. Such an approach leads to a truly device and application independent standard graphic system and allows device independent programming with full usage of the workstation capabilities.

3.6 Attributes

3.6.1 Attributes for output primitives

Surprisingly markers in GKS are treated differently from pen or text. For the latter two a pen or text number is used as an entry in the workstation pen or text attribute description table, whereas for the marker its only attribute must be set directly, e.g. marker size. A marker number should be provided in GKS.

3.6.2 Workstation attributes

Significantly, the GKS document states that the workstation attributes may be different for different workstations. According to the discussion in section 3.5, the setting of workstation attributes should not be part of the standard graphic system. However, the setting of those attributes which are obviously common to all applications might remain, e.g. thickness and colour of lines, text font.

3.7 Storage of graphical information

3.7.1 Short term storage

As suggested in section 3.3., the segment storage should be removed from GKS.

3.7.2 Long term storage

It is not evident that a unique standard format for transporting and storing graphical information can be widely acceptable. Although related to the content of a standard graphic system, such a format (or formats) should be the subject for a separate standard. However, the proposed GKSM certainly needs to be reviewed.

GKS level concept

As many of the levels defined in GKS are based on features which have been refuted in the previous sections, e.g. segment insert, segment storage, pixel array, they will not be discussed here. However, it can be suggested to separate output and input capabilities.

CONCLUSION

Following is a summary of the most important issues arising from this review which, we think, are worth considering while designing a standard for graphics. Needless to say we acknowledge and appreciate the tremendous and valuable work which the GSPC and the GKS people have put into the production of the CORE and GKS proposals.

i) The viewing system should be separated from the basic graphic system and the latter be standardized first.

ii) The primitive PIXEL ARRAY should be removed from the standard graphic system.

iii) The NDC should be replaced by a user defined coordinate system with metric units.

iv) The segment concept should be reviewed in light of the separation between the viewing system and the basic graphic system.

v) The insert segment function and the related segment storage should be removed from the standard graphic system.

vi) The setting of non common workstation attributes should be removed from the standard graphic system.

vii) A standard format for storing and transporting graphical information should be left to a separate standard.

REFERENCES

/1/ DIN 00 66 252 : GKS graphical kernel system version 5.2.

/2/ Graphics Standard Planning Committee : Status report, Computer Graphics Vol. 11, number 3, fall 1977.

/3/ Graphics Standard Planning Committee : Status report, Computer Graphics Vol. 13, number 3, August 1979.

/4/ GUEDJ, R.A. et al., Report on the IFIP W.G. 5.2. Workshop on "Methodology in Computer Graphics" (July 1976), North Holland Publ.

/5/ GRAVE M., OUANOUNOU G., "Basic Objects and Operators for Computer Animation". Proceedings Eurographics, 1979.

ANNEX 1

PROPOSAL OF STANDARD DIN 00 66 252

INFORMATION PROCESSING
GRAPHICAL KERNEL SYSTEM (GKS)
TERMINOLOGY

In the description of the Graphical Kernel System (GKS), commonly accepted computer graphics terminology is used as far as possible. This clause gives the definition of the important terms.

1 attribute:
A particular property that applies to a display element (output primitive) and a display group.
Example: Low intensity, green colour, blinking status.
Note: In GKS this definition is also used for workstations.

2 choice device:
An input device providing integer numbers specifying alternatives.

3 clipping (see scissoring)

4 coordinate graphics; line graphics:
Computer graphics in which display images are generated from display commands and coordinate data.

5 cursor:
A movable, visible marker used to indicate a position on a display space.

6 device coordinate (DC):
A coordinate expressed in a coordinate system that is device dependent.
Note: In GKS this definition is restricted to cartesian coordinates measured in meter, used for specifying the display space.

7 device driver:
The device dependent part of a GKS implementation intended to support a graphics device. The device driver generates device dependent output and handles device dependent interaction.

8 display device; graphics device:
A device (cathode ray tubes, plotter, etc.) on which display images can be represented.

9 display element; graphic primitive; output primitive:
A basic graphic element that can be used to construct a display image.
Example: A dot, a line segment, a character
Note: The element can itself be the complete display image.

10 display group; segment:
A collection of display elements that can be manipulated as a unit and that can be further combined to form larger groups.
Note: In GKS this definition is restricted to an ordered collection of display elements (output primitives) defining a display image.

11 display image; picture:
A collection of display elements or display groups that are represented together at any one time on a display space.

12 display space; operating space; workstation viewport:
That portion of a display surface available for a display image.
Note: The display space may be all or part of the display surface.

13 display surface:
In a display device, that medium on which display images appear.

14 GKS level:
A number describing the set of functional capabilities provided by a specific GKS implementation.

15 GKS metafile (GKSM):
A sequential dataset that can be written and read by GKS, used for longterm storage of graphical information.

16 graphics device (see display device)

17 graphic primitive (see display element)

18 highlighting:
Emphasizing a display element or a display group by modifying its visual attributes.
Note: In GKS this definition is restricted to display groups (segments).

19 input device class:
One of the following five types of input primitives: locator, valuator, choice device, pick device, string device.

20 input primitive:
A basic graphic element from an input device (such as keyboard, function keys, joy stick, control ball, or a light pen)

21 line graphics; coordinate graphics:
Computer graphics in which display images are generated from display commands and coordinate data.

22 locator:
An input device providing a coordinate position as its result.

23 marker:
A glyph with a recognizable appearance which is used to identify a particular location.

24 normalized device coordinates (NDC):
Device-independent cartesian coordinates in the range 0 to 1 used for specifying the viewport and the workstation window.

25 operating space (see display space)

26 output primitive (see display element)

27 pen table; colour table:
A list associating predefined pen number, linetype, linewidth, and colour with a pen number.

28 pick device:
An input device to identify a display element or a display group.

29 pick identifier:
A number for identifying primitives within a segment by the pick input function. The same pick identifier can be assigned to different primitives.

30 picture (see display image)

31 picture element; pixel:
The smallest element of a display surface that can be independently assigned a colour or intensity.

32 polyline:
A display element (output primitive) consisting of a set of connected lines.

33 polymarker:
A set of markers.

34 raster graphics:
Computer graphics in which display images are generated on a display surface composed of a matrix of pixels arranged in rows and columns.

35 rotation:
Turning all or a part of a display image about an axis perpendicular to the display surface.
Note: In GKS this definition is restricted to display groups (segments).

36 scale:
Enlarging or reducing all or part of a display image by multiplying the coordinates of that image by a constant value.
Note: For different scaling in two (three) orthogonal directions two (three) constant values are required.
Note: In GKS this definition is restricted to display groups (segments).

37 scissoring; clipping:
Removing parts of a display image that lie outside a boundary, usually a window or viewport.
Note: Clipping guarantees to include all parts of the display elements that lie within the boundary where scissoring does not.

38 segment (see display group)

39 segment attributes:
Attributes that apply only to segments: visibility, highlighting, detectability, segment priority, and segment transformation.

40 segment transformation:
A transformation which causes the display elements defined by a segment to appear at varying positions (translation), sizes (scale), and/or orientations (rotation) on the display surface.

41 shift (see translation)

42 string device:
An input device providing a character string as its result.

43 text table:
A list associating string attributes with a text number.

44 translation, shift:
the application of a constant displacement to the position of one or more display elements.
Note: In GKS this definition is restricted to display groups (segments).

45 valuator:
An input device providing a real number as its result.

46 viewing transformation (see window/viewport transformation)

47 viewport:
A predefined part of the display space.
Note: In GKS this definition is restricted to a rectangular region within the normalized device coordinates used for the definition of the window/viewport transformation.

48 window:
A predefined part of the virtual space.
Note: In GKS this definition is restricted to a rectangular region within the world coordinates used for the definition of the window/viewport transformation.

49 window/viewport transformation; viewing transformation:
A transformation that maps the boundary and contents of a window to the boundary and interior of a viewport.
Note: In GKS this transformation maps positions in world coordinates to normalized device coordinates.

50 workstation:
A console (station) that includes a display device and/or also one or more input devices such as an alphanumeric keyboard, function keys, a joy stick, a control ball, or a light pen.

51 workstation transformation:
A transformation that maps the boundary and contents of a workstation window to the boundary and interior of a workstation viewport (display space).
Note: In GKS this transformation maps positions in normalized device coordinates to device coordinates.

52 workstation viewport (see display space)

53 workstation window:
A rectangular region within the normalized device coordinates system which is represented on a display space.

54 world coordinates (WC):
Device independent cartesian coordinates used by the application program for specifying graphical primitives and transformations.

55 zooming:
Progressively scaling the entire image to give the visual impression of movement towards or away from an observer.
Note: In GKS this definition is restricted to display groups (segments).

ANNEX 2

PROPOSAL OF STANDARD DIN 00 66 252

INFORMATION PROCESSING

GRAPHICAL KERNEL SYSTEM (GKS)

FUNCTIONAL DESCRIPTION

1. Scope of Application and Purpose

This standard specifies a set of functions for graphical data processing in a way which is independent from particular graphic devices, programming languages, or applications.

The capabilities provided by this standard include:

- two dimensional line and raster graphics
- graphical input and output primitives at one or more graphic workstations
- provision for storage and modification of one set of graphical information in a workstation independent manner during program execution
- storage and retrieval of graphical information from a long term graphics file (Metafile)
- means for adapting the application program behaviour to suit workstation capabilities
- several upward compatible levels of the standard with increasing functional capabilities

This standard has emerged from a long process of refinement. In this process of adapting the GKS system to ISO recommendations and in order to reduce the gap between GKS and other similar international standards under development, some parts of the standardisation concepts of ISO, ANSI, and ACM-SIGGRAPH (GSPC) have been included in the GKS proposal. A few details have been taken over literally.

2. Related Standards

DIN 44 300 (ISO 2382)	Informationsverarbeitung; Begriffe (information processing; vocabulary)
DIN 66 003 (ISO 646-7.73)	Informationsverarbeitung; 7-Bit-Code (information processing; 7-bit-code)
DIN 66 004 (ISO 962-11.74)	Informationsverarbeitung; Darstellung des 7-Bit-Code auf Datenträger (information processing; 7-bit-code representation on storage media)
DIN 66 010	Magnetbandtechnik fuer Informationsverarbeitung; Begriffe (magnetic tape technique for information processing; vocabulary)

3. The Graphical Kernel System (GKS)

GKS contains two dimensional output and input primitives. The output primitives include line drawing, text drawing, and raster graphics primitives. Five classe of input primitives (LOCATOR, VALUATOR, PICK, CHOICE, STRING) are supported which can work in REQUEST, EVENT, and SAMPLE mode.
A segment facility allows subdividing pictures into subparts. Segments may be created and deleted and the segment attributes can be dynamically modified. The segments can be transformed and inserted into other segments.
The graphical output can be routed to one or multiple workstations. A workstation is a console that includes one display device and/or also one or more input devices such as an alphanumeric keyboard, function keys, a joy stick, a control ball, or a light pen. The coordinates are transformed in a two-stage transformation process where the first stage can be set for each primitive and the second stage can be set for each workstation. Furthermore the setting of a workstation specific pen and text table allows to control the appearance of all primitives on the corresponding workstation.

The GKS Metafile serves for long term storage of graphical data. It also provides a standard interface between GKS and other graphics systems.
Not every GKS implementation has to support the full set of functions. Six GKS levels have been defined to meet the different requirements of graphical systems.

GKS defines only a language independent nucleus of a graphics system. For integrating it into a language, GKS should be embedded in a language dependent layer containing the language conventions, e.g. parameter and name assignment.

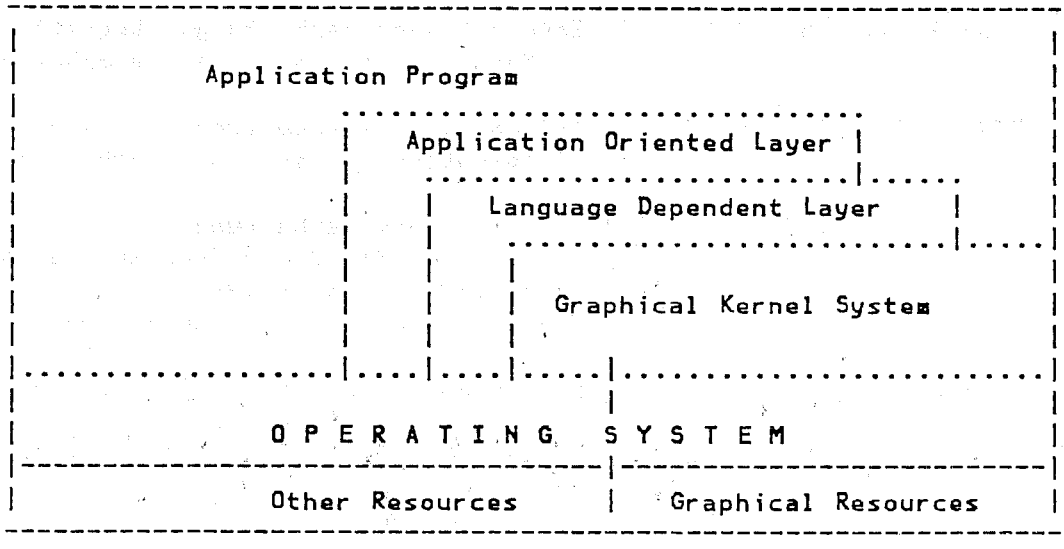

Figure 1: Layer Model of GKS

The layer model represented in figure 1 illustrates the application of a GKS in a graphical system. Each layer may call the functions of the lower layers. In general the user program will use the application oriented GKS layer, the language dependent layer, other application dependent layers, and operating system resources. All workstation capabilities that can be addressed by GKS functions must only be used via GKS.

3.1 Graphical Output

The graphical information that is generated by GKS and routed to all active workstations is built up of basic pieces called ouput primitives. GKS provides three line drawing primitives, one text primitive, and two raster graphics primitives:

POLYLINE	GKS generates a polyline defined by a point sequence.
POLYMARKER	GKS generates centered symbols at given positions.
DRAW	General function to address special capabilities of a workstation like drawing of spline curves, circular arcs, and elliptic arcs. The objects are characterized by an identifier and a set of points. GKS only applies all transformations to the points leaving the interpretation to the workstation.
TEXT	GKS generates a character string at a given position.
FILL AREA	A polygon is filled with a uniform colour.
PIXEL ARRAY	An array of pixels with individual colours is drawn.

Each primitive comprises

- the geometrical information, text data, and raster information as specified in the individual function call (what is to be presented)
- the attribute information which - in conjunction with the corresponding control setting of the workstation - controls how the primitives are to be presented on a specific work station (see chapter 3.3 and 3.6).

3.2 Graphical Input

3.2.1 Input Classes and Types

Graphical input can be obtained from any open workstation. The graphical input functions provided by GKS can be grouped into five classes specifying the kind of input primitives and three input types that depend on how the input is obtained from the workstation.

The five input classes are:

LOCATOR	provides a position in world coordinates
VALUATOR	provides a real number
CHOICE	selects an alternatives
PICK	provides a segment name and a pick-identifier
STRING	provides a character string

All five classes of input can be obtained from a workstation by use of three different mechanisms. The three input types are:

REQUEST: GKS reads a tuple of input primitives of one input class from the workstation. GKS waits until the input is entered or an end-of-input action is performed.

SAMPLE: GKS inspects the current setting of an input device and delivers back the current value without waiting for an operator action.

EVENT: GKS builds up one input queue into which input primitives from various sources are lined up in the sequence in time in which they are generated. The queue can be inspected by the application program in order to find out whether or not input primitives are present and from which source they originated.

For REQUEST type input GKS provides five functions, one for every input class, e.g. REQUEST TUPLE OF LOCATORS. Besides the selection of a specific input device, the maximum number of input is specified as a function parameter. The operator at the workstation has the possibility to enter input primitives of the requested class. GKS waits until the requested number of inputs have been entered or an end-of-input action is performed by the operator. The REQUEST input behaves in this way similar to a FORTRAN READ.

For SAMPLE type input the input devices have to be enabled (e.g. ENABLE VALUATOR). The current value of the input device can be sampled (e.g. SAMPLE VALUATOR) and immediately transferred back to the application program. If the hardware allows testing of the current setting of the device, the sampled value is derived from this setting, as is the case for VALUATOR input by use of a potentiometer. In other cases the last setting or the undefined value set by e.g. ENABLE VALUATOR is passed back as the sampled value.

For EVENT type input one event queue is present in GKS. After one or more input devices are enabled (e.g. ENABLE PICK), the workstation operator has the possibility to enter input primitives into the queue. Two functions serve for testing the oldest element of the queue: READ EVENT and AWAIT EVENT. In the case the queue is not empty, both functions behave in the same manner. They deliver the input class and the input device identification of the oldest element back to the application program. Then the value of the tested element can be obtained by use of a GET-function, e.g. GET PICK. If the queue is empty, AWAIT EVENT waits until an input is entered while READ EVENT returns to the calling program immediately reporting that the queue is empty.
Two further functions are provided for handling input originated by a

specified workstation: READ WORKSTATION EVENT and AWAIT WORKSTATION EVENT. They behave in the same manner as READ EVENT and AWAIT EVENT with the restriction that they inspect only the subset of the queue originated by the specified workstation leaving unchanged entries in the queue originated by other workstations.
FLUSH functions are provided for deletion of the input primitives of a given class or all input primitives in the input queue.
An input device can be either enabled for SAMPLE or for EVENT type input. REQUEST type input can be obtained only from disabled devices.

3.2.2 Prompting and Echoing

Prompting and echoing are input device characteristics that can be controlled by the application program. An echo of the input can be switched on and off (SET ECHO) and its position can be specified (SET ECHO POSITION). SET LOCATOR and SET VALUATOR are functions for initializing LOCATOR and VALUATOR input devices. These functions affect the values returned by the SAMPLE type input, if the setting is not changed after the initial calls.

The function SET CHOICE PROMPT sets the prompting of a CHOICE input device. The input device that are most commonly used to implement the CHOICE input functions often have built-in prompting capabilities (e.g. lamps on function keyboards). The SET CHOICE PROMPT function allows the application program to invoke this prompting capability.
By the function SET CHOICE STRINGS a text string can be associated with every alternative of a CHOICE input device. These text strings may be used for several purposes:

- Prompting or Help:
  Each text string gives a keyword or short description of the corresponding alternative. The text string can be displayed above the functions keys on the display surface or can be used to explain the meaning of the alternatives via a help function.
- Modelling of a CHOICE device:
  The driver has the possibility to create a string menu at an implementation dependent location from which the operator can choose the desired alternative.
- Keyboard as a CHOICE device:
  The operator can type in the textstring of the desired alternative. In this case a corresponding CHOICE device number must be predefined for the workstation.

The function ASSIGN SEGMENT TO CHOICE allows the application program to define a segment that has been created previously as a menu for choice class input. After a segment has been assigned to a CHOICE, it will be activated by using a pick device passing back the pick identifier of the indicated primitive as choice value.

All coordinate values for input functions (e.g. LOCATOR input, SET ECHO LOCATION, SET LOCATOR) are expressed in world coordinates. If

normalized coordinates are desired the application program must set the window/viewport transformation to identity before calling these functions.

3.3 The Segment Concept

In GKS, graphical output primitives can be generated outside segments or they may be grouped in segments identified by a segment name.
Primitives outside segments are sent to all active workstations, the application program has no access to them any more after they have been generated.
Segments may be

- deleted
- renamed
- made visible and invisible
- made detectable or undetectable
- highlighted
- transformed
- inserted into the open segment

Only primitives contained inside segments are affected by these operations. Furthermore, the INSERT function applies only to segments created while a (virtual) segment storage workstation has been activated.
Every primitive within a segment has a pick identifier associated with it which establishes a second level of naming. The sole function of it is the identification of primitives, it cannot be used for manipulations. This level of naming has been introduced to reduce the segment overhead for applications where a great number of picture parts should be distinguishable and where the manipulation possibility is less important.

```
SET PICK IDENTIFIER (1)
  Output functions                          not pickable
CREATE SEGMENT  (1)
  Output functions          ---> segment=1, pick-id.=1
SET PICK IDENTIFIER (2)
  Output functions          ---> segment=1, pick-id.=2
CLOSE SEGMENT (1)
  Output functions                          not pickable
CREATE SEGMENT  (2)
  Output functions          ---> segment=2, pick-id.=2
SET PICK IDENTIFIER (1)
  Output functions          ---> segment=2, pick-id.=1
SET PICK IDENTIFIER (2)
  Output functions          ---> segment=2, pick-id.=2
CLOSE SEGMENT (2)
```

Figure 2: Example for the use of segment names and pick identifiers.

Whereas segment names must be assigned uniquely the pick identifier can be assigned arbitrarily to single output primitives or groups of output primitives within segments.

When a segment is closed, primitives cannot be modified nor can primitives be added to or deleted from the segment. No function is provided to extend a segment after it has been closed. However, geometrical transformations, changes of the segment attributes and of the workstation specific pen and text tables are possible.
The segment attributes (VISIBILITY, HIGHLIGHTING, DETECTABILITY, and SEGMENT PRIORITY, see chapter 3.6.2) describe the segment as an entity. All values describing the state of a segment, i.e. name, segment attributes, and workstations active at creation time, are stored in a segment state list that GKS keeps during a segment's lifetime.

Every segment is stored on all workstations active at the time it is created (OPEN SEGMENT). It can be deleted on all workstations by the DELETE SEGMENT function All segments stored on a specific workstation can be deleted by the DELETE ALL SEGMENTS function.
Furthermore, segments as an entity can be transformed by the TRANSFORM SEGMENT function i.e. scaled, rotated, and translated (in exactly this sequence). The segment transformation is stored and applied to the segment before displaying it.

Segments stored on a special workstation called segment storage workstation can be inserted under transformation into the open segment by the INSERT SEGMENT function. When inserting a segment into the open segment, the specified transformation is carried out and the transformed segment is copied into the open segment.

3.4 Coordinate Systems and Transformations

GKS regards three levels of coordinate systems (see figure 3):

- One world coordinate system (WC)
- One normalized device coordinate system (NDC)
- Device coordinate systems (DC)

Two transformations are applied to an output primitive routed from the application program to the display surface:

- A window/viewport transformation that maps the world coordinate space onto the NDC space,
- A workstation transformation that is used to map the NDC space onto the DC space individually for every workstation.

The application programmer defines all graphical output primitives in two dimensional cartesian coordinates called world coordinates. By means of the window/viewport transformation the graphical output primitives are mapped from WC onto normalized device coordinates, they

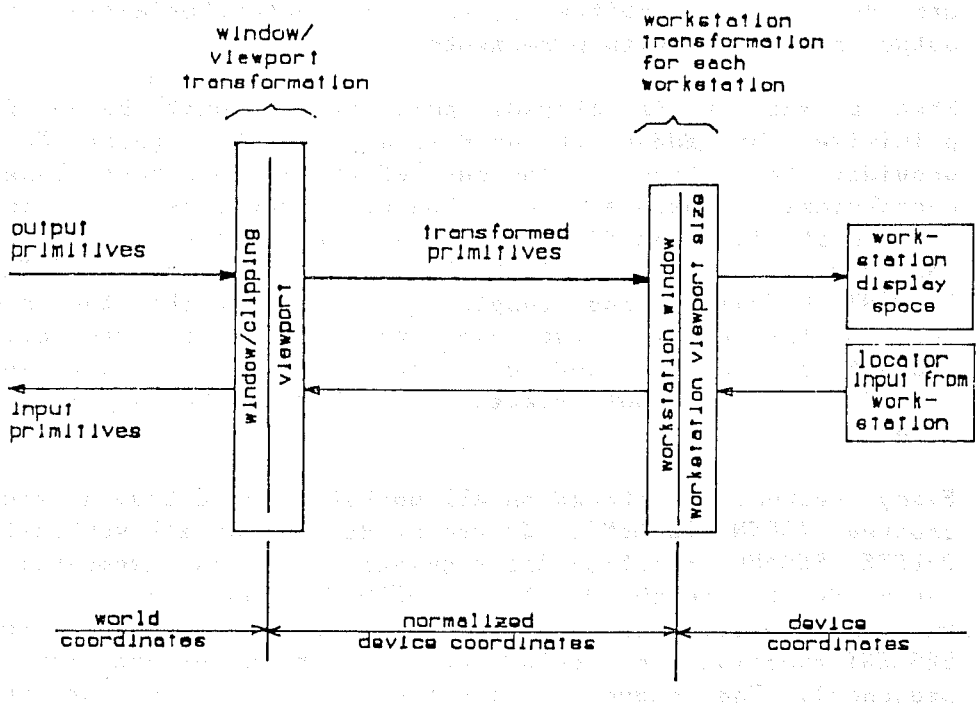

Figure 3: Coordinate Systems and Transformations

are in the range (0,1)x(0,1). The window/viewport transformation is applied to every output primitive individually. It can be reset at any time after GKS has been opened. The window is specified by two corner points of a rectangle parallel to the coordinate axes in world coordinates. The viewport is specified by the two corresponding points in NDC.
The normalized device coordinate system will be mapped onto the device coordinate system, that describes the display space of a workstation. The device coordinates are measured in meter. This workstation transformation can be set individually for every workstation, allowing the same graphical primitive on different display surfaces in different , application program controlled, scales. In this way it is possible, e.g., to use the full display space of an interactive workstation and to have simultaneously a drawing in correct scale on a plotter. The workstation transformation is set by specifying a workstation window in NDC and the workstation viewport size in DC space.

The default setting of the workstation transformation will map one of the sides of the unit NDC square onto the shorter side of the display space. The same workstation transformation is valid for all graphical output primitives on an individual workstation. A resetting of this transformation changes the scale of all segments stored on the respective workstation.

The current values of the window/viewport transformation are kept in the GKS state list, a basic data block that reflects the state of GKS. The settings of the workstation transformation are contained in a workstation state list present for every open workstation.
The window/viewport transformation may be accompanied by clipping at

the window which can be switched on and off by the application program. Even if clipping is switched off GKS takes care that the result of the window/viewport transformation does not exceed the unit square and that the result of the workstation transformation does not exceed the display space.
Locator input data is transformed back by the inverse workstation transformation valid when LOCATOR input is generated and by the inverse window/viewport transformation valid when LOCATOR input is read by the user program.

The segment transformations (TRANSFORM SEGMENT, INSERT SEGMENT, see figure 4) are a third type of transformation provided by GKS. They apply to segments only and map NDC onto NDC. However, parameters are specified by the application program in WC and transformed into NDC by the current window/viewport transformation. By this concept a GKS user always specifies coordinates in WC space with the exception of the viewport parameter for the window/viewport transformation and the parameters for the workstation transformation.
Segment transformations are not actually performed in the segment storage but only saved in the segment state list. In this way a segment transformation cannot lead to a loss of information.
After the segment transformation is applied to primitives within a segment the workstation transformation will be applied to them. In an implementation all transformations may be united to a single transformation matrix.

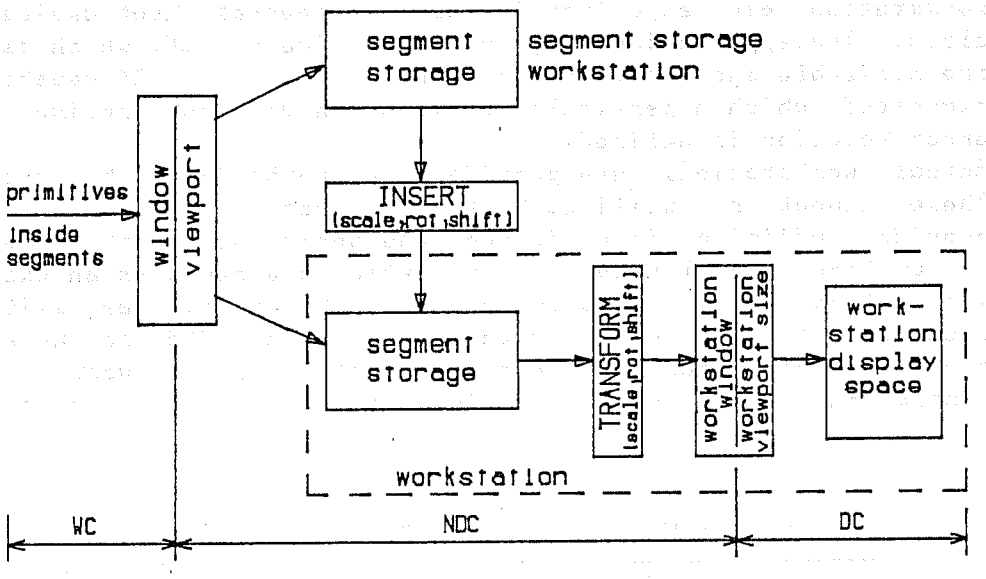

Figure 4: Segment Transformations

3.5 The Workstation Concept

3.5.1 Workstation Characteristics

GKS is based upon a concept of an abstract graphical workstation with capabilities as described below. For every kind of workstation present in a given GKS implementation (e.g. refresh display, storage tube, plotter), an entry exists in a workstation description table. It describes the capabilities and characteristics of the workstation. For every open workstation its state is kept by GKS in a workstation state list.
The fully equipped abstract workstation

- has one addressable display surface
- permits the use of smaller display spaces than the maximum while guaranteeing that no display is generated outside the specified display space. An installation dependent ("paper" or "background") quality may be specified.
- supports several linestyles, text fonts, character sizes, etc.
- has one or more input devices for each class of input primitive.
- permits request type, event type, and sample type input.

In actual installations the workstation may or may not be equipped with all of these capabilities. E.g., the input capabilities of a workstation may range from no input to several input devices of every class. The application program may inquire via GKS which capabilities are available and adapt its behaviour accordingly. If capabilities are requested which a particular workstation does not provide, a standard error reaction is defined.
Actual workstations may provide more capabilities than those listed. These cannot be utilized by GKS. However, if the workstation itself provides sufficient intelligence, the additional capabilities may well be utilized locally by the workstation operator. As an example, if a workstation has two display surfaces, the operator may switch locally from one to the other without notifying GKS or the application program. More than one display surface at one workstation can be controlled by GKS only by defining a separate work station for every display surface.

3.5.2 Selecting a Workstation for Output and Input

The workstations are identified by the application program by use of a workstation identifier. Output primitives are sent to all active workstations. Segment manipulation and input can be performed with all open workstations. Input devices at a particular workstation are identified by the tripel: workstation identifier, input class, input device number. The latter selects one of the input devices of one class at a given workstation. E.g. a workstation may have a lightpen and a tablet for locator class input. For EVENT type input the input device at a given workstation must be enabled.

```
OPEN WORKSTATION (N1, ddname1, workstation type1)
OPEN WORKSTATION (N2, ddname2, workstation type2)
ACTIVATE WORKSTATION (N1)
  Output functions                    ---> generated on N1
  REQUEST input            input allowed only from N1,N2
  SAMPLE input             input allowed only from N1,N2
ACTIVATE WORKSTATION (N2)
  Output functions              ----> generated on N1,N2
DEACTIVATE WORKSTATION (N1)
  Output functions                  ----> generated on N2
  ENABLE (N1,class,nr)
  EVENT input              input allowed only from input
                      device 'nr' of class 'class' on N1
  DISABLE (N1,class,nr)
CLOSE WORKSTATION (N1)
DEACTIVATE WORKSTATION (N2)
CLOSE WORKSTATION (N2)
```

Figure 5: Example for selecting workstations for input and output

3.5.3 Deferring Picture Changes

The display of a workstation should always reflect the actual state of the picture as defined by the application program. To use the capabilities of a workstation efficiently, it may be desirable to allow a workstation to defer the actions requested by the application program by calling GKS functions for a certain period of time. During this period the state of the display may be undefined. E.g., data sent to a plotter may be blocked to optimize data transfer.
All functions leading to an implicit regeneration of the whole picture on a workstation may be suppressed until a regeneration is required explicitly. An implicit regeneration is necessary, e.g., when picture changes require erasing the display of a storage tube workstation or to put new paper on a plotter. When functions are invoked, that may require an implicit regeneration, all primitives outside segments will be deleted from the display surface.
The function SET DEFERRAL STATE allows to choose that deferral state which takes into account the capabilities of the workstation and the requirements of the application program.

The following deferral states can be specified:

State 1: The visual effect of each function has to become visible as soon as possible.
State 2: The visual effect of each function has to become visible before the next interaction (i.e. no deferring if an input device is enabled; update before 'ENABLE input primitive' or 'REQUEST input primitive' is executed).
State 3: The visual effect of all functions may be deferred.
State 4: As state 1 but implicit regeneration is suppressed.
State 5: As state 2 but implicit regeneration is suppressed.
State 6: As state 3 but implicit regeneration is suppressed.

	Deferral states					
	1	2	3	4	5	6
Addition of graphical data	o	i	x	o	i	x
Implicit regeneration	o	i	x	1	1	1

Legende: o - no deferring
i - may be deferred until input
x - may be deferred
1 - must be deferred

Figure 6: Deferral states

The deferring applies to following functions:

Functions generating output:
POLYGON, POLYMARKER, TEXT, DRAW
FILL AREA, PIXEL ARRAY
INSERT SEGMENT
READ AND INTERPRET NEXT RECORD FROM GKSM
Functions that may imply an implicit regeneration:
SET PEN/TEXT REPRESENTATION
SET WORKSTATION WINDOW/VIEWPORT
SET VISIBILITY/DETECTABILITY/HIGHLIGHTING/SEGMENT PRIORITY
DELETE SEGMENT, TRANSFORM SEGMENT

The concept of deferring refers only to visible effects of GKS-functions. Effects on the segment storage or in the state of the workstation are (conceptually) not deferred.
There is no state provided where the addition of graphical data must be deferred. Such requirements should be handled with the segment storage facility and the visibility attribute. By this restriction the buffer for deferred actions can be chosen implementation dependent.
Deferred actions can be made visible at any time by the use of the UPDATE-command or by appropriate change of the deferral state.

3.6 Attributes

3.6.1 Attributes for Output Primitives

- PEN NUMBER, an index referring to an entry in a pen or colour table in which pen attributes are specified.
- TEXT NUMBER, an index referring to an entry in a table in which text attributes are specified.
- MARKER SIZE specifying the size of centered symbols.
- PICK-IDENTIFIER for identifying primitives in segments when the segment is picked.

The current value of these attributes can be set modally and is recorded in the GKS state list. During generation of an output primitive this value is copied into the primitive and cannot be changed afterwards. PEN NUMBER and PICK-IDENTIFIER apply to all output primitives, TEXT NUMBER applies only to the TEXT primitive, and MARKER SIZE only to the POLYMARKER primitive.

3.6.2 Segment Attributes

- VISIBILITY (a segment is displayed or not)
- DETECTABILITY (a segment can be selected by the pick input primitive or not)
- HIGHLIGHTING (a segment is blinking or not)
- SEGMENT PRIORITY (if parts of segments (pixel array, filled area) overlap, the segment with higher priority will be displayed)
- SEGMENT TRANSFORMATION (a segment is transformed before display)

The segment attributes are unique for each segment and may not vary on different workstations. Each segment is assigned the default values of these attributes when the segment is opened. The attributes may be reset for the open segment and for any existing segment.
The segment transformation has been described in chapter 3.4.

3.6.3 Workstation Attributes

- PEN REPRESENTATION, pen table (also referred to as colour table) every entry containing: predefined pen number, linetype, linewidth, intensity, colour(red/green/blue intensity)
- TEXT REPRESENTATION, text table, every index containing: predefined text number, text font, text quality, character size and spacing.
- DISPLAY SPACE QUALITY, e.g. kind of paper on a plotter.
- DEFERRAL STATE, controlling the deferred or immediate visibility of picture changes.
- WORKSTATION TRANSFORMATION (see chapter 3.4)

These workstation attributes control the appearance of output primitives on a specific workstation, the quality and size of the display space, and the immediate visibility of primitive generation and picture manipulation.
The representation of a symbolic PEN or TEXT NUMBER on a specific workstation is defined via a pen or text table in the workstation state list. Some standard definitions for table entries are contained in the workstation description vector and are used as default. The application program may set the tables by copying a standard definition or by specifying explicitly the attributes of a specific entry.

The workstation attributes may be different for different workstations

and may be changed.
If the change affects the display of segments already stored on the workstation this will be called a dynamic attribute change. This feature is demanded only in GKS level 4 and is intended for workstations having appropriate facilities (e.g. raster displays with colour table).
Otherwise the resetting of work station attributes is only allowed if no graphical data is present on the workstation e.g. immediately after OPEN WORKSTATION or RESET WORKSTATION (static attribute change).
The deferral state is explained in more detail in chapter 3.5.3.

3.7 Storage of Graphical Information

3.7.1 Short Term Storage

As already pointed out in chapter 3.3 the segment manipulation facility requires the storage of all segments for reuse on the same workstation. GKS does not prescribe the manner and format of storage as long as all segment operations can be performed. For the purpose of transferring segments from one workstation to another or for inserting segments into the open segment a (virtual) segment storage workstation is used, where segments are stored in a device independent way for the purpose of the INSERT function. Whether the segment storage workstation is realized within the GKS nucleus or by utilizing the capabilities of an appropriate physical workstation is left to the implementer.

3.7.2 Long Term Storage

For the purpose of long term filing of graphical information GKS provides an interface to a sequential file called GKS Metafile (GKSM). It can be used for:

- transportation of graphical information between systems,
- transportation of graphical information from one place to the other (e.g., by means of magnetic tapes)
- storage of graphical information from one GKS-application to the next one.
- storage of accompanying non-graphical information.

The GKSM behaves like a workstation. It can be thought of as simulating a passive output device (e.g. plotter). For output to the GKSM and input from it two different workstations with appropriate workstation sequence numbers must be defined. In order to preserve consistency of the file contents, the GKSM may only be written and read under GKS control. For this purpose special functions are provided. Figure 7 shows the relationsship between the application program, GKS and the GKSM.

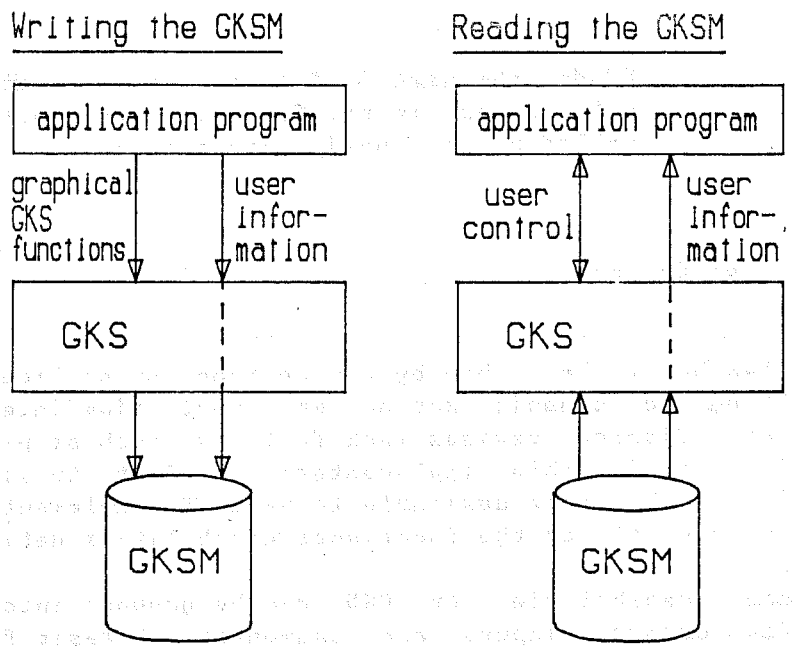

Figure 7: Relationship between GKS and GKSM

Input from the GKSM is processed by a special GKS input driver under the control of the application program. Reading and interpreting a record from the Metafile is not corresponding to an input function, but it has the same effects as if the GKS function was called, that previously generated this GKSM record. E.g. reading and interpreting a polyline record will cause the same effects as if the polyline function is called. In order to be able to reconstruct graphical objects from the GKSM, besides primitives also attributes and state information must be stored on the Metafile.

Following functions are related to the GKSM:

Output to GKSM:
- OPEN/CLOSE WORKSTATION
- ACTIVATE/DEACTIVATE WORKSTATION
- All output functions
- Attribute setting functions

(applied to a GKSM output workstation)

- WRITE USER INFORMATION TO GKSM
  Passes user data to the GKSM

Input from GKSM:
- OPEN/CLOSE WORKSTATION
- ACTIVATE/DEACTIVATE WORKSTATION

(applied to a GKSM input workst.)

- READ RECORD TYPE FROM GKSM
  Passes the type and length of the next GKSM record back to the application program
- READ RECORD FROM GKSM
  Passes the next GKSM record to the application program (graphical or user record)
- SKIP NEXT RECORD ON GKSM
  Reads and ignores the next GKSM record
- READ AND INTERPRET NEXT RECORD FROM GKSM

Reads the next GKSM record and generates the same effects as if the functions contained in it were called by the application program.

3.8 GKS Level Concept

The GKS system has to be usable by a wide range of applications, from static plotting to dynamic motion and real time interaction. In addition, many display devices lack features (such as picking) that would require considerable implementation effort to simulate with software. It is therefore desirable to have GKS implementations that must not include all of the functional capabilities defined in this standard.
The functional capabilities of GKS can be grouped into four major areas: basic, output, input, and segmentation. Basic features are those that are essential for any GKS implementation. The areas output, input, and segmentation can be subdivided further based on levels of complexity, treatment of attributes and capabilities of the input and output devices and the segment storage mechanism.

If an arbitrary combination of capabilities was considered a valid GKS implementation, an almost unlimited number of different standard dialects would result and one of the major goals of this activity would not be achieved - that is program portability. Therefore six valid levels of the GKS system have been defined, in order to address the most common classes of equipment and applications:

The indentation in the following table has the meaning:

- Capability present
  - Capability partially present
    - Capability missing

Level 0: (Minimal capabilities)

- Minimal workstation control
  - No setting of workstation attributes
  - Only one workstation at a time
- Minimal output
  - No raster output, no DRAW function
- Output to a metafile workstation possible
  - No metafile input and no user data
  - No input
  - No segment mechanism

Level 1: (Passive output)

- Workstation control
  - Static change of workstation attributes
- Full output
- All metafile functions
  - No input
  - No segment mechanism

Level 2a: (Passive output and segment storage, no input)
- Workstation control
  - Static change of workstation attributes
- Full output
- All metafile functions
  - No input
- Segments stored for TRANSFORM and INSERT
  - Segment attributes highlighting, visibility, priority
    - No pick identifier

Level 2b: (Passive output and simple input, no segments)
- Workstation control
  - Static change of workstation attributes
- Full output
- All metafile functions
  - Request type input
    - No pick input
    - No segment mechanism

Level 3: (Passive output, simple input, and segment mechanism)
- Workstation control
  - Static change of workstation attributes
- Full output
- All metafile functions
  - Request type input
    - No pick input
- Segments stored for TRANSFORM and INSERT
  - Segment attributes highlighting, visibility, priority
    - No pick identifiers

Level 4: (Full set of GKS capabilities)
- Full workstation control
- Dynamic change of workstation attributes
- Full output
- All metafile functions
- Request, sample, and event type input
- Pick input included
- Segments stored for TRANSFORM and INSERT
- All segment attributes including detectability

A complete listing is given in chapter 8.5. It must be pointed out, that the level concept refers to a GKS implementation, not to the features of one of the workstations connected to it. Of course, at least one workstation must be able to realize the functional capabilities of a GKS implementation.
In figure 8 the functional capabilities present with each GKS level are summarized.

<table>
<tr><th>level / feature</th><th>0</th><th>1</th><th>2a</th><th>2b</th><th>3</th><th>4</th></tr>
<tr><td>basic features</td><td colspan="6">window/viewport transformation
minimal control | full control, workstation transformation inquiry, GKSM</td></tr>
<tr><td>output</td><td>no raster
no DRAW
no attr. setting</td><td colspan="4">full output
static attribute change</td><td>dynamic</td></tr>
<tr><td>segmentation</td><td></td><td></td><td>segments not detectable</td><td></td><td>segments not detectable</td><td>all segment attributes</td></tr>
<tr><td>input</td><td></td><td></td><td></td><td colspan="2">request input
no pick</td><td>full input</td></tr>
</table>

Figure 8: GKS level concept

3.9 The States of GKS

Five different operating states may occur in the GKS (see figure 9):

0 = GKS closed;
1 = GKS open;
2 = At least one workstation open;
3 = At least one workstation active;
4 = Segment open

These operating states differ for the user in so far as individual calls to GKS are allowed only in certain operating states as indicated in the functional description in chapter 4. At each instance between two calls of GKS the overall state of GKS is defined by a set of state variables having specific values. These state variables are characterized by the fact that they allow a complete description of the effects of the functions. The total set of GKS state variables contains the following subsets:

- GKS state list
- Segment state list for every existing segment
- Workstation state list for every open workstation
- GKS error state list

On the basis of individual functions these state subsets are allocated, made available and cancelled.
When allocating these state subsets they will be initialized with

default values. When initializing a workstation state list, some of the default values will be taken from a workstation description table where an entry is present for every kind of workstation supported by a GKS implementation. The variables of the state subsets are modified by the GKS functions and can be queried by the application program.

When an error condition is detected during execution of a GKS function, GKS calls an error procedure. During execution of the error routine GKS is in an error substate which corresponds to the operating state in which the function was called upon. In this error substate GKS allows only inquiry functions but no modifications to any of the state lists except the error state list.

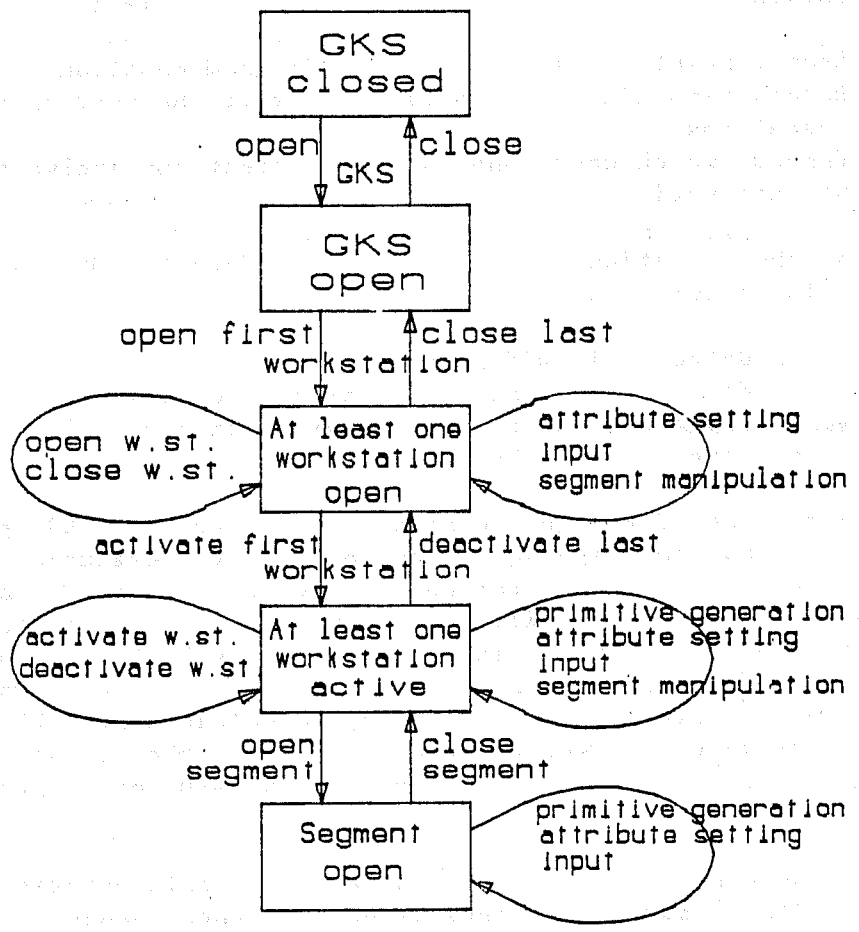

Figure 9: Possible Transitions between Operating States

3.10 Error Handling

For each GKS function a finite number of error situations is listed which will cause the error handling routine to be called. The name of the error handling routine is supplied to GKS by the application program. The error handling routine (unless a standard one is used) provides an interface between GKS and the application program. The error handling routine may interpret the information about the error (supplied by GKS in the error state list) and may store data in a data area for subsequent interpretation by the application program after return from the erroneously called GKS function.
The GKS error handling strategy is derived from the following error classification:

- I Errors resulting in a precisely defined reaction
- II Errors resulting in an attempt to save the results of previous operations
- III Errors which cause unpredictable reactions including the loss of information

Regarding the location where an error is detected GKS distinguishes the following situations:

- A Error detected in GKS procedures
- B Error detected in procedures called from GKS (driver procedures, operating system procedures)
- C Error detected in other areas of the application program

If errors are detected outside GKS (situation C) either the application program may regain control over the execution or program execution will be terminated abnormally. In the latter case results are unpredictable (case III) and in the worst case all graphical information produced so far in this job may be lost. If, however, the application program obtains control it may attempt to close GKS properly or at least attempt an emergency closure by calling the GKS emergency closure procedure. Similarly, if the error occurs in procedures called by GKS and control is not returned properly to GKS, effects are unpredictable.

The GKS emergency closure procedure is an implementation dependent facility. Its purpose is to save as much of the graphical information produced as possible. The effects of this procedure on the work stations are left undefined in this standard. The emergency closure procedure may be called directly from the application program. It is also called from GKS itself as a standard error reaction. Errors of this type belong to class II.

Finally, all errors which are listed explicitly as part of the definition of GKS functions belong to class I. They are either detected within GKS itself (situation A) or a procedure called from GKS (situation B) has returned control to the corresponding GKS procedure with the appropriate error information. In all these cases of class I GKS calls the error handling procedure whose name is passed

to GKS when it is opened. The user may either provide his own error handling procedure or may use the standard error handling procedure provided as part of GKS. Any error handling procedure must have access to the following information:

- The identification of the GKS function which calls the error procedure
- The identification of the error condition
- A record which is passed from the GKS function to the error handling procedure, containing all supplementary information regarding the error condition.
- A record which belongs to the address space of the application program and which may be used to exchange information between the application program and the error handling procedure.

The standard GKS error handling procedure performs the following actions:

- Print an error message on the error message file
- Return to the calling program

Example of an user supplied error handling procedure

```
Errorhandling: PROCEDURE
     Interprete GKS procedure and error identification in order to
     select the following cases:
- CASE "special treatment":
     Interpret GKS error record; Store information for application
     program in user data record; Return to calling GKS procedure;
- CASE "standard treatment":
     Call standard GKS error handling procedure with all above
     parameters; Return to calling GKS procedure;
```

All GKS procedures after detecting an error condition perform the following actions:

- Store procedure identification, error identification, and supplementary error data into the GKS error state list.
- Set error state.
- Call error handling procedure.
- Reset error state.
- Perform built-in error reaction.

The user supplied error handling procedure has access to all accessible GKS state table information in accordance with GKS operating state prior to the GKS function call which caused the error. However, no modification of GKS state is possible during error handling. I.e. only GKS inquiry functions may be called by the error handling procedure. This is achieved by setting and resetting the error state indicator prior and after calling the error procedure from GKS.