RWH

RAL-89-097

Science and Engineering Research Council
Rutherford Appleton Laboratory
Chilton DIDCOT Oxon OX11 0QX

RAL-89-097

Second Generation Computer Graphics Standards

KW Brodlie, F R A Hopgood and D A Duce

September 1989

Second Generation Computer Graphics Standards

K.W. Brodlie‡, F.R.A. Hopgood† and D.A. Duce†

‡School of Computer Science, University of Leeds, Leeds, U.K.

†Informatics Department, Rutherford Appleton Laboratory, Chilton, Didcot, OXON OX11 0QX, U.K.

1. The First Generation

Standardization activities have existed in computer graphics since the early 1970's and a family of standards for computer graphics is now emerging from the International Organization for Standardization/ International Electrotechnical Commission (ISO/IEC). This family of standards covers a broad range of graphics requirements from application program interfaces for the generation and interactive manipulation of 3D graphics, to device level interfaces for the transfer of graphical information.

The major standards in progress are:

(1) **GKS** (Graphical Kernel System) - a set of basic functions for 2D device-independent computer graphics programming.

(2) **CGM** (Computer graphics metafile for transfer and storage of picture description information) - a device independent data exchange format for computer graphics pictures.

(3) **CGI** (Interface techniques for dialogues with graphical devices) - a set of basic elements for the control and data exchange between device-independent and device-dependent levels in graphics.

(4) **GKS-3D** (Graphical Kernel System for 3 Dimensions) - an extension of GKS to provide the basic functions for computer graphics programming in 3D.

(5) **PHIGS** (Progammer's Hierarchical Interactive Graphics System) - a set of functions for computer graphics programming in environments requiring rapid modification of graphical data that describes geometrically related objects.

(6) **Language Bindings** - bindings of the functions and data types of the functional standards to standardized programming languages.

(7) **Registration** - a registration mechanism is being set up to deal with the standardization of primitive aspects, generalized primitives, escape functions and other graphical entities.

(8) **Conformity Testing of Implementations of Graphics Standards** - the purpose of this project is twofold: first to specify the characteristics of standardized test sets for use in determining the conformance of implementations of graphics standards and second to provide direction to developers of functional standards concerning conformance rules.

In order to describe the status of the various standardization projects, it is necessary to define the various stages a standards document goes through within ISO/IEC before becoming an international standard. These are:

(1) *Workitem.* An official project with agreed scope and goals and timescales. When an area has been identified for standardization, a proposal for a project is prepared. There is a ballot on the proposal within the appropriate joint technical committee (JTC) and, if successful, the workitem is assigned to a particular Subcommittee, who manage the project and assign it to a particular Working Group (WG) to carry out the technical work. The Subcommittee for computer graphics is SC24, within JTC1, designated JTC1/SC24. The WG structure of SC24 is outlined below. A rapporteur (manager) and document editor are appointed for the project by the SC.

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

(3) *Draft International Standard (DIS).* When sufficient agreement is reached on the DP document, the revised document is registered as a DIS. The publication of a DIS should indicate that technical agreement has been reached. The document is then circulated within the JTC for review and ballot. Comments submitted are addressed and resolution sought. Any remaining disagreements at this stage can cause another DIS ballot, but normally the document proceeds to the next stage.

(4) *International Standard (IS).* The DIS revised in the light of comments received becomes the Final Text. A final ballot within ISO/IEC Council ensures that all members are satisfied that the ISO/IEC procedures have been followed by the project. Eventually the standard is published and the text is available from ISO/IEC Central Secretariat, or through national standards bodies. It is common for international standards to be issued as national standards, under a national number, possibly in translation. Standards are reviewed 5 years after publication, at which time they may be endorsed, revised or abandonned.

(5) *Addenda.* A mechanism exists for enhancing a standard prior to the five year review. This is done by publishing an Addendum to the standard. Addenda are progressed through similar phases to standards themselves.

The computer graphics subcommittee, SC24, has five working groups:

(1) **WG1: Architecture.** Charged with developing a computer graphics reference model, solicitation of user requirements in the area of computer graphics and currently with the revision of GKS.

(2) **WG2: Application Program Interfaces.** Standardization of functional specifications for application program interfaces.

(3) **WG3: Metafiles and Device Interfaces.** Standardization for graphical information exchange, including computer graphics metafile and device interfaces.

(4) **WG4: Language Bindings.** Standardization of language bindings for functional standards.

(5) **WG5: Validation, Testing and Registration.** Development of methods and procedures for testing and validation of implementations of computer graphics functional standards and development of methods and procedures for the registration of graphical items.

Voting is a slow process and consequently the development of standards takes a long time. The documents tend to be long and intricate and the commenting process demands a large amount of, usually, volunteer labour. The status of the projects referred to above is shown in table 1.

Project	Ref Doc	Availability of text for: WD	DP	DIS	IS
GKS	ISO7942				IS (1985)
GKS Language bindings					
Fortran	ISO8651-1				IS(1988)
Pascal	ISO8651-2				IS(1988)
Ada	DIS8651-3				IS(1988)
C	DP8751-4		6/88		
GKS-3D	DIS8805				IS(1988)
GKS-3D Language bindings					
Fortran	DIS8806-1				4/89
Pascal	SC24/N190	7/88			
Ada	SC24/N189	7/88	2/89	6/90	
C	SC24/N181	7/88	3/89	12/89	
PHIGS					
Functional description	DIS9592-1				IS(1989)
Archive file format	DIS9592-2				IS(1989)
Archive file clear text encoding	DIS9592-3				IS(1989)
PHIGS Language bindings					
Fortran	DIS9593-1		7/87	11/87	12/88
Extended Pascal	DP9593-2		5/89	12/89	
Ada	DIS9593-3		9/87	1/89	8/89
C	DP9593-4		5/89	12/89	
CGM					
Functional description	ISO8632-1				IS (1987)
Character encoding	ISO8632-2				IS (1987)
Binary encoding	ISO8632-3				IS (1987)
Clear text encoding	ISO8632-4				IS (1987)
CGM Addendum 1					
Functional description	ISO8632-1/ADD.1			10/88	10/89
Character encoding	ISO8632-2/ADD.1			10/88	10/89
Binary encoding	ISO8632-3/ADD.1			10/88	10/89
Clear text encoding	ISO8632-4/ADD.1			10/88	10/89
CGM Addendum 2					
Functional description	N23	3/88	5/89	10/89	10/90
Character encoding	N24	10/89			
Binary encoding	N25	10/89			
CGI	DP9636		11/88	10/89	2/91
CGI Character encoding	SC24/N209	3/89	10/89		
CGI Binary encoding	SC24/N210	3/89	10/89		
CGI Library language binding					
Fortran	SC24/N192	1/89	8/89	4/90	6/91
C	SC24/N191	1/89	8/89	4/90	6/91
Conformity Testing	SC24/N185	8/88	3/89	12/89	10/90

Table 1

1.1. GKS

The Graphical Kernel System (GKS) was published as an ISO Standard on 15 August, 1985. The Standard itself is defined in.[8] A more detailed introduction and primer are given in[6] while a full and comprehensive treatment may be found in.[5] The following sections briefly describe the key concepts in GKS.

1.1.1. Dimensionality

GKS is a two-dimensional graphical system and provides no support for three dimensions. The extension of GKS to 3D is described later.

1.1.2. Primitives

The six basic output primitives are polyline, polymarker, fill area, text, cell array and generalized drawing primitive (GDP). The polyline primitive draws a set of lines between a sequence of points. The polymarker primitive is similar but marks the sequence of points with a specified symbol. The text primitive provides considerable flexibility in defining the quality of the text, its size and orientation, the origin etc. It also supports the text path being in any of the major directions providing support for those languages not writing from left to right. The fill area primitive is defined in terms of a set of points which specify a polygon. The primitive fills the enclosed area with a solid colour or a specified pattern or hatch style.

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

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

The GKS output primitives have a rich set of aspects, allowing a high degree of control over the way primitives are rendered on displays. The aspects of a polyline primitive, for example, allow control over the linetype (solid, dotted etc.), linewidth and colour. The mechanisms by which the values of aspects are determined are described shortly.

1.1.3. Coordinate Systems and Device Independence

The GKS concept of a workstation is the key to device independence in GKS. A workstation consists of zero or one display surface and zero or more input devices plus associated software. The GKS idea of a workstation is an abstraction from physical hardware.

A major difference from many earlier graphics systems is that GKS allows more than one workstation to be in use simultaneously. For example, an operator may be interacting with a design through an interactive display, while taking copies of completed parts of the design on a plotter.

Output primitives are specified in a cartesian world coordinate system. Applications that require other user level coordinate systems, for example, polar or logarithmic coordinates, must first transform these user coordinates to world coordinates.

Transformation to the coordinate system of the display device is accomplished in two stages. First, world coordinates are transformed to an intermediate coordinate system called normalized device coordinates (NDC) by a window to viewport mapping termed a normalization transformation. Then a second window to viewport mapping, the workstation transformation, transforms these coordinates to device coordinates.

The purpose of the normalization transformation is to facilitate the composition in NDC space of pictures defined in different world coordinate system spaces. Different device coordinate systems are accommodated in the workstation transformation. Thus to use an application program with different devices, it is only necessary to change the workstation transformation. The composition of the picture in NDC space does not need to be changed.

The workstation transformation may be set differently for different workstations, thus allowing different regions of the virtual picture to be displayed on different workstations. Through the workstation activation and deactivation mechanism, not all primitives need be displayed on all workstations.

The aspect ratios of window and viewport may differ in the normalization transformation, but the workstation transformation maps the workstation window to the largest possible region of the workstation viewport with the same aspect ratio.

1.1.4. Aspects

The appearance of primitives on the display surface of a workstation is controlled by their aspects. GKS distinguishes two types of aspects, workstation independent aspects which have the same value on all workstations on which the primitive is displayed, and workstation dependent aspects which may have different values on different workstations.

The values of aspects are controlled by attributes. For workstation independent aspects, there is one attribute per aspect. These attributes are termed geometric attributes. For workstation dependent aspects, two methods of specification are possible, bundled specification and individual specification. Bundled specification uses a lookup table approach. A single attribute for each primitive, the primitive index, controls the values of all the workstation dependent aspects of the primitive.

The polymarker primitive will be used as an example. The values of all the aspects of a polymarker (marker type, markersize scale factor and polymarker colour index) are determined by the value of the polymarker index attribute. A polymarker index defines a position in a table, the polymarker bundle table. Each entry in this table specifies values for all the non-geometric aspects of a polymarker. Each workstation has its own bundle table and so a polymarker in the virtual picture may be displayed with different representations on different workstations.

In the case of individual specification, there is one attribute for each workstation dependent aspect and thus each aspect has the same value on each workstation on which the primitive is displayed. How each workstation approximates this value is dependent on the workstation and the implementation.

A set of aspect source flags controls the mode of specification of each aspect. Some aspects may be specified individually, whilst others are specified by a bundle.

Bundled specification is a powerful tool for achieving application program portability between different workstation environments. If carefully constructed, moving a program to a different environment will merely mean defining new representations for the different indices used in the picture, to employ, in the best way possible, the characteristics of the workstations in the new environment. The bundled scheme is important when it is necessary to ensure that primitives with different attributes can be differentiated on different workstations; whilst the individual scheme is important when primitives with specific aspects are to be represented on each workstation as closely as possible to the specification.

1.1.5. Graphical Input

A major innovation in GKS is the model of input. The aim was to specify a set of virtual input devices onto which real input devices could be mapped.

All input devices are formalized as having a measure and a trigger. The measure describes the type of input value returned by the device, while the trigger is the event which causes the measure value to be returned to the application program in certain styles of input.

The data that can be entered into an application program by the operator are divided into six different types, and six classes of logical input device are defined corresponding to these. The data types are:

LOCATOR: a position in world coordinates and the associated number of the normalization transformation used to convert back from device coordinates via NDC to world coordinates. The normalization transformation used is that whose viewport contains the data-point. Conflicts are resolved by a priority mechanism.

STROKE: similar to LOCATOR except that it represents a sequence of world coordinate positions rather than a single position.

VALUATOR: a real number in some range.

CHOICE: an integer that represents a selection from a set of choices.

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

STRING: a character string.

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

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

SAMPLE: the current measure value is returned whenever requested by the application program. The trigger is not used by SAMPLE input.

EVENT: a number of input devices may be active together. Each time the trigger for a particular device is activated, the current measure value and data that identify the device are added to a single queue of input events for all the devices used in event mode. The application program can interrogate the queue to retrieve the input events. It is possible to couple more than one input device to the same trigger so that multiple events can be generated from a single trigger event. Unfortunately, this coupling is not under application program control.

Some degree of control over logical input devices is provided to the application program through device initialization functions. These enable the program to define the initial value for the device, the prompt/echo type (for example a LOCATOR device may be echoed as a rubber band line, tracking cross etc.), the area of the display to be used for displaying the echo, and further device dependent data.

1.1.6. Segments

Associated with each workstation is a segment store in which segments consisting of sets of GKS primitives and associated attributes can be stored. Functions exist to create, delete, rename and manipulate segments. Associated with each segment is a set of attributes which control visibility, highlighting, priority for output and detectability from a pick device. It is also possible to transform segments such that the picture defined by the segment can be scaled, rotated, translated etc.

There is also a workstation independent segment storage which is used as a central library. Additional operations are available for segments stored in this storage, for instance to enable them to be moved to other workstations.

1.1.7. Levels

Rather than insist that all facilities in GKS are supported by every implementation, GKS is defined as a set of levels on two orthogonal axes, output and input. There are nine levels in total, ranging from level 2c which includes everything, to level 0a which has no input facilities and only simple output facilities.

1.2. GKS-3D and PHIGS

The two functional standards that extend GKS to 3 dimensions are GKS-3D and PHIGS. GKS-3D is a minimal extension to GKS to allow 3D working. GKS-3D provides application programs with the capability to define and display 3D graphical primitives specified using 3D coordinates. The GKS input model is also extended to provide 3D locator and stroke input. A major goal in the design of GKS-3D was that existing GKS programs should run, as far as possible, without change.

Neither GKS or GKS-3D satisfy the requirements of application programs where modification of the graphical data is required in an efficient manner, where the objects to be displayed consist of geometrically related parts and where rapid dynamic articulation of graphical entities is required. PHIGS aims to address applications with these requirements. The main features of these two systems are now described.

1.2.1. Storage

The segment store in GKS-3D is identical to that in GKS except for the extension to 3 dimensions. Segments are normally stored on workstations and there is a centralized workstation independent segment store to allow movement of segments from this storage to other workstations as required.

The major difference between PHIGS and GKS is that in PHIGS the creation and display of a picture are very explicitly independent phases. At the heart of PHIGS is a single centralized structure store (CSS), which has greater functionality than the GKS-3D segment store and appears at a different (higher) place in the viewing pipeline.

A structure consists of a number of structure elements which can be both graphical and non-graphical. Thus it is possible to keep application data associated with graphics in the same database. PHIGS provides facilities for creating and editing structures held in the CSS.

PHIGS provides a set of functions which define structure elements. There is a 1-1 correspondence between primitives and structure elements. In addition, PHIGS has structure elements for attribute setting and specification of application data. A major feature of CSS is that it is hierarchical. Structure elements are provided which call other structures.

Structures are displayed when they are posted to a workstation. Posting causes the structure to be traversed and interpreted. A complex modelling transformation is applied to coordinates in the structure elements as they are interpreted and an equally complex modelling clipping operation may also be applied.

The primitives generated by PHIGS enter the viewing pipeline immediately prior to the viewing operation. The modelling transformation provides a similar function to the normalization transformation and the modelling clip represents a generalization of the normalization clip.

1.2.2. Structures

Particular features of the PHIGS structure facility are:

(1) *Hierarchy.* Structures can call other structures and the same structure may be called more than once from a higher level. Thus a car may need only a single wheel structure which is called four times (five including the spare!).

(2) *Modelling coordinates.* Structure elements contain positional information in modelling coordinates, a cartesian coordinate system. Each structure has a global and local modelling transformation which are concatenated to produce the transformation to be applied to the points to turn the modelling coordinates into the world coordinates to be passed to the viewing pipeline. Modelling clipping regions may be defined in modelling coordinates as the intersection of a collection of half-spaces. These regions are then transformed to world coordinates and combined, using standardized or implementation dependent combinators, to produce a composite modelling clipping region which may be applied to graphical output.

(3) *Inheritance.* Substructures inherit attributes, modelling transformation and modelling clipping from the calling structure. Thus the global modelling transformation for a structure will be that passed in by the calling structure. When a structure has been completely traversed, control reverts to the higher structure that called it and the attributes, transformations etc. are reset to those in force on entry to the substructure. Thus a substructure has no effect on the calling structure.

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

1.2.3. Viewing

Viewing consists of projecting the 3D image onto a 2D projection plane. The viewing pipeline is given in Figure 1. Functions are provided to assist with the definition of this viewing operation. The initial coordinates are changed to Viewing Coordinates by defining a View Reference Point and a set of axes associated with it. The intention is that this point has some relationship to the object to be viewed and makes the setting up of the projection transformation that much easier.

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

Each primitive has a View Index associated with it which defines the view bundle table entry on the workstation to be used. This contains details of the viewing transformation and clipping to be applied. It was believed that, unlike GKS, there is a need for more than one view to be available at a time on a workstation. This would, for example, allow titles to be output using a parallel projection while a 3D object to which the titles are associated is output using a perspective transformation. The view bundle is analogous to the polyline bundle in that it allows views to be different on different workstations.

Support is provided for Hidden Line and Hidden Surface removal in both GKS-3D and PHIGS. Associated with primitives is an attribute defining which method of rendering is to be used on the workstation. The workstation can be asked to render or not and it has flexibility in how it does the rendering. Consequently, a variety of workstations can choose the most appropriate methods depending on their hardware characteristics.

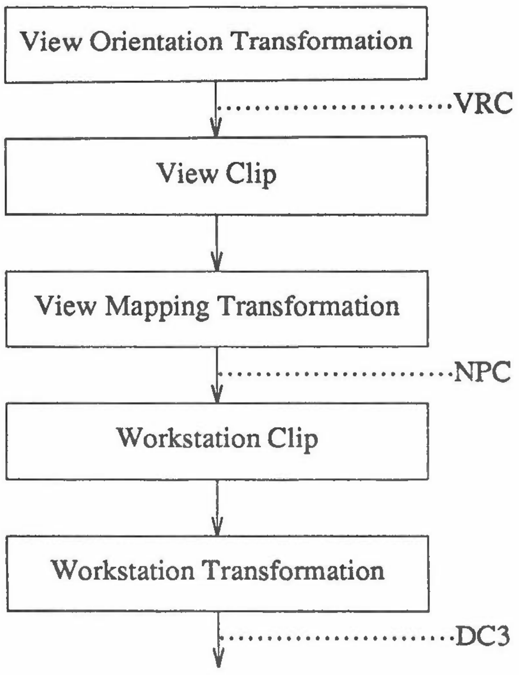

View Orientation Transformation
VRC
View Clip
View Mapping Transformation
NPC
Workstation Clip
Workstation Transformation
DC3

Figure 1

1.2.4. Primitives

GKS-3D and PHIGS provide the same basic primitive set as GKS except that they are extended to work in the 3D environment. Existing GKS functions can be called an produce the equivalent GKS-3D primitive on the Z=0 plane (effectively a Z=0 coordinate is added to each position in a GKS function call).

In GKS-3D and PHIGS, text, fill area and cell array remain planar primitives, but can be positioned in an arbitrary plane. Polyline and polymarker become genuine 3-dimensional primitives with no constraints on the positions used in the function call.

One additional primitive has been added to both GKS-3D and PHIGS, FILL AREA SET, which specifies a set of fill areas all of which will be rendered together as a single entity. For rendering 3-dimensional objects, it was believed that this extension was necessary. For the same reason, FILL AREA SET has more control on how the boundary edge is rendered than the original fill area primitive. There are some subtle, annoying, differences between the rules governing the rendering of the primitive in GKS-3D and in PHIGS.

A further primitive has been added to PHIGS, ANNOTATION TEXT RELATIVE. The purpose of this primitive is to facilitate labelling of objects. The primitive is defined in NPC space, its position is defined by a reference point in modelling coordinate space and an offset from this in NPC coordinate space. The plane on which the annotation appears is always parallel to the x-y plane of the display space and is unaffected by modelling and viewing transformations, but the reference point is transformed normally. Height, orientation, path and alignment attributes may be specified for the primitive in NPC space and an annotation style may also be specified which defines how the relationship between the reference point and the text string will be displayed. The options available for the latter include unconnected, lead line displayed with current polyline aspects and implementation dependent manner. This primitive is useful in certain contexts, but it remains to be seen to how large a class of applications this will extend.

In GKS and GKS-3D, eligibility of primitives for picking, visibility and highlighting are determined by segment attributes. PHIGS has no segment store and rather than carry over the GKS analogy through a structure attribute mechanism, it was recognized that more flexible ways of controlling such entities are necessary in a system designed to serve the needs of highly interactive applications. All primitives in PHIGS have a NAME SET attribute. This is an addition to the GKS and GKS-3D attribute sets. The nameset attribute defines the eligibility of the primitive for highlighting, invisibility, picking and incremental spatial search (a ''software'' search which can be performed on the centralized structure store). A filter mechanism is incorporated in PHIGS with an inclusion and exclusion set of names at each workstation to control specific rendering of primitives. Primitives whose name sets have at least one name in common with the inclusion set and no names in common with the exclusion set of a particular filter have the property (highlighted, invisible, pickable etc.) controlled by that filter. This gives a very flexible mechanism for structuring the models defined by the centralized structure store. For example, the heating system or electrical system or water system of a house could be selectively displayed by merely changing filters, if the primitives corresponding to objects in these systems have appropriate name set attributes.

It was felt that this mechanism would be useful in GKS and GKS-3D also, but it has been decided that the mechanism should not be incorporated in GKS-3D at this stage, but is to be considered in the 5 year review of GKS.

1.2.5. Input

The input models in all three standards are very similar. Both GKS-3D and PHIGS extend the logical input devices by allowing 3-dimensional locator and stroke devices as well as the six logical input devices defined in GKS.

Pick input in PHIGS has been extended to give more information about what has been picked. In GKS and GKS-3D, the PICK device returns the name of the segment and the PICK identifier within the segment. As a structure in PHIGS could be executed as part of one or more parent structures, some applications will need to know more than just the local structure name. Consequently, in PHIGS it is possible to recover the names of the structures in the hierarchy that led to the invocation of the structure that has been picked.

1.2.6. Colour Models

Colour models define a colour coordinate space and a subspace, in which each point represents a describable colour. GKS only supports the RGB colour model, in which colours are described by triples of values ranging from 0 to 1 or 0% to 100% for each of the three primaries.

PHIGS and GKS-3D allow other colour models to be used, including CIE, HSV and HLS. The application program can select the colour model in which the tuple of values specifying colours will be interpreted. The CIE colour model was introduced in response to the observation that ''device independent'' colour is becoming a major issue in computer graphics. The specification of colour is an area in which there is increasingly close collaboration between the makers of graphics standards and the makers of standards for document production and printing.

1.2.7. Summary

This section has only given an informal introduction to the new facilities available and the differences between PHIGS and GKS-3D. Stringent efforts have been made to harmonize PHIGS and GKS-3D and remaining differences either reflect different requirements for their respective fields of application or fundamental differences of opinion about the conceptual model by which they are related.

1.3. Language Bindings

In the early days of ISO work on graphics standards, it was realized that the different languages from which people used graphics made it necessary to define the functional specification in a way that was independent of any one language. The functional standards, GKS, GKS-3D and PHIGS therefore define a collection of functions and data types which are intended to be sufficiently abstract as not to hinder binding to any particular programming language. There is a separate ISO standardization activity to cover language bindings for GKS, GKS-3D and PHIGS. Bindings for these standards to FORTRAN, Pascal, Ada and C are either completed or in preparation. Table 1 shows the current status of the language binding standards. The rules of the game are that language bindings may only be standardized for programming languages which themselves have some status as standards. Thus there are no standardized bindings to languages such as Prolog or Lisp, though there is much interest in bindings to these languages and alternative approaches continue to be reported in the literature.[7] The paper by Sparks and Gallop[10] gives a good overview of the problems and approaches to language binding standardization.

1.4. The Computer Graphics Metafile (CGM)

GKS provides a mechanism for the storage of graphical information, segment storage, but this only provides a method for the storage of transient information and is not designed for long-term storage between sessions. Once the workstation is closed, segment storage for that workstation ceases to exist.

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

As it became clear that there would be more than one graphics standard at the functional level and all would have a need for long term storage and retrieval of graphical information, it was decided to separate out the metafile function as a separate standard. Thus was born the Computer Graphics Metafile (CGM) for the storage and transfer of picture description information (ISO 8632) which was approved for progression to an International Standard in September 1986.

GKS does include a set of functions for reading and writing metafiles and an Annex which specifies a metafile of the audit trail type which is adequate for the needs of GKS. The Annex (E) is not an intrinsic part of the standard, but if provided by an implementation, will allow communication between GKS systems or long-term storage and auditing within a GKS system.

GKS Annex E describes an audit trail metafile in which the entire process of creating a picture is stored for future replay. Essentially the metafile records every function invoked in the creation of the picture. In contrast, the CGM is a picture capture metafile and captures a snapshot of the graphical image. It is probably fair to say that at the time the CGM project was started, the differences between these two types of metafile were not widely or deeply understood and harmonization problems have arisen as a result. The CGM metafile does not include any elements which imply dynamic change to the image, nor does it record any segmentation structure to the image. This can cause problems for GKS applications which wish to write metafiles to the CGM standard. The relationship between the CGM and GKS is explored in the paper by Brodlie, Henderson and Mumford.[4]

GKS-3D poses even more profound problems in this area as a result of the much more complex transformation pipeline and satisfactory compromises are currently being sought.

For a detailed description of the CGM, see the book by Arnold and Bono.[3]

1.4.1. Functional Specification of the CGM

Each instance of the CGM is a collection of elements. Figure 2 summarizes the overall structure of a metafile as a series of levels.

(1) *Metafile level.* A metafile consists of a metafile descriptor and picture descriptions. The metafile descriptor contains information that is valid for the whole metafile, for example the precision of real and integer quantities in the metafile. The main body of the metafile contains descriptions of a number of independent pictures that can be accessed individually. These descriptions are self-contained in that the description of one picture is not dependent on any information stored in another picture definition.

(2) *Picture level.* Pictures descriptions are bounded by particular delimiters. Between these delimiters is a picture descriptor which defines how the picture definition data are stored. The description of the picture itself is contained within the picture body.

(3) *Picture body level.* The elements at this level are essentially primitive and attribute elements describing the graphical content of the picture.

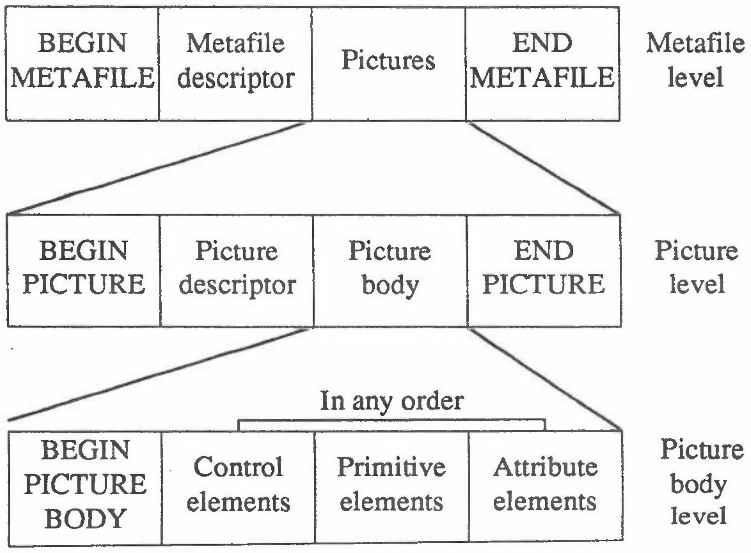


Figure 2

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

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

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

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

(4) *Graphical Elements.* These describe the visual components of the picture being transferred. CGM includes more primitives than GKS, for example, circular arcs, disjoint polylines and rectangles.

(5) *Attribute Elements.* These specify the attributes of the graphical elements.

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

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

1.4.2. Encodings of the CGM

The first part of the standard describes the elements that may appear in a metafile and the constraints on their ordering. Subsequent parts of the standard define the representations for these elements in actual metafiles. Three representation schemes, known as encodings, have been standardized. Each has particular goals such as compactness, ease of generation and interpretation, ease of transfer across networks. The three encodings are:

(1) *Character Encoding.* The aims of this encoding are compactness and transferrability across a network. The encoding is composed solely from the ISO 7-bit (ASCII) printing characters. Elements are encoded as an opcode followed by associated data.

(2) *Binary Encoding.* This encoding aims to minimize the processor effort required to generate and/ or interpret the metafile. It is perhaps best suited for storage and retrieval of graphical data within one system.

(3) *Clear Text Encoding.* This encoding is aimed at the requirement of having a metafile that can be read and edited by people. It is almost guaranteed that this format can be used for transfers between any pair of heterogeneous systems.

CGM also allows private encodings as long as they conform to the functional description and general rules of conformance given in Part 1 of the Standard.

1.4.3. Conformance

It is worth noting that the conformance statements in CGM relate to the conformance of a metafile. They do not refer to the processes of generating or interpreting metafiles. Thus there can be no guarantee that a metafile written by one generator can be understood by every interpreter. Commercial products typically differ in the range of CGM elements they generate and the range they can interpret, though the description of each metafile does contain a list of element types used in that metafile, so one can *a priori* determine whether a given metafile can be interpreted by an interpreter with given capabilities. This limited conformance requirement needs to be borne in mind when purchasing software purporting to offer CGM.

1.4.4. Implementations

By the end of 1987 over two dozen companies in the USA had released products or announced plans for products incorporating support for the CGM. Three kinds of product are emerging: applications such as CAD or business graphics packages may offer the ability to write a CGM file capturing one or more of the pictures created during a design session, second applications such as desktop publishing systems or printer spoolers which can read in CGM files and make some use of the pictures contained therein (for example paste into document or print on device), and third a small number of applications (such as graphics editors) can take in a CGM picture description, manipulate it and write it out again as a CGM description.

A major boost was given to CGM in the USA at the Integrate '88 demonstration at the NCGA '88 conference and exhibition held in Anaheim in March 1988. Integrate '88 was a multivendor systems integration demonstration which incorporated four application areas typically found in a multifaceted corporation: engineering/design, corporate communications/financial analysis, graphics arts and computer-aided publishing. Some 38 vendors took part in the demonstration. Multi-vendor equipment and software were linked together through an ethernet communications infrastructure running TCP/IP and used CGM as the standard picture interchange format. The demonstration was organized as a series of scenarios. In a typical scenario an operator would retrieve a design created on a CAD program. The CAD representation was then modified and output as a file in CGM format. In this format it was then sent to the graphics art department for enhancement or directly to printing and publishing for merging with text in a brochure. Similar procedures were used to exchange files between finance, graphics arts, and printing and publishing. It was an impressive demonstration of the potential of interchange formats such as CGM in promoting product harmonization and is worth mentioning for that reason alone.

A second demonstration of this type was held at the Eurographics UK Chapter Conference in Manchester in March 1989 where more encodings were interchanged. The NCGA '89 demonstration included PHIGS as part of the interchange system for the first time.

1.4.5. CGM Addenda

As mentioned above, CGM in its present form does not include all the facilities necessary to serve as a GKS metafile for all levels of GKS. A project known as CGM Addendum 1 is addressing this and reached the status of draft in December 1987. The principal elements added in Addendum 1 include:

(1) segmentation support;

(2) capabilities needed for dynamic picture regeneration;

(3) device viewport control.

Some stable functionality from CGI is also being included for example closed figures and pixel array and drawing mode support.

A second Addendum, CGM Addendum 2, is being processed to provide support for GKS-3D by introducing appropriate 3D elements.

1.5. The Computer Graphics Interface (CGI)

The Computer Graphics Interface is an interface to graphics devices. It is intended as the interface through which a device driver communicates with a device. Unlike the CGM which defines the output from a graphics system for transmission or storage, the CGI standard has to handle both output and input and it is assumed that the device is on-line and capable of supporting dynamic interactive graphics. The CGI has to support a wide range of devices from simple plotters to high powered interactive terminals and it is this diversity which is one of the main reasons why CGI has been under development for so long and agreement still looks to be some way away (although it has now reached DIS stage). The book by Arnold and Bono[3] gives a comprehensive account of the CGI draft as it existed at the end of 1987.

CGI is a multipart standard with six parts. Part 1 is an overview introducing the other parts as follows.

(1) Part 2 - control, negotiation and errors. Control functions are provided for device management and coordinate space specification. Device management includes initialization and termination, deferral mode control etc. Negotiation is the process of establishing the capabilities of the device the driver will use. This involves interrogation of the facilities provided by the device and selection of those to be used. Error handling is necessarily different to functional standards and CGI provides the ability to turn off error reporting and detecting.

(2) Part 3 - output and attributes. The output primitives provided in CGI are very similar to those in CGM. Bundled and individual specification of aspects are supported.

(3) Part 4 - segmentation. CGI defines two types of stored graphics object, segments and bitmaps. The segment model is close to the GKS model and the operations provided over segments are essentially the operations inherent in GKS. A mechanism is provided to enable primitives stored in a segment to acquire new attribute values when a segment is copied. This is an extension of the rather bizarre feature of GKS by which primitives acquire a new clipping rectangle when certain segment manipulation operations are performed.

(4) Part 5 - input. The CGI input functionality is designed to support the input models of the functional standards. CGI provides the six input classes of GKS together with an area class and general input class.

(5) Part 6 - raster. This functionality allows the creation, storage, manipulation and display of images stored as sets of pixels. The bitmap functionality is not found in other graphics standards. Bitmaps provide a second point in the CGI pipeline at which graphics output data can be stored and modified, at the point where primitives have been rendered to pixel values. A bitmap can be selected as the destination for graphics output, allowing portions of a picture to be defined and named. Bitmaps can be combined using either two or three operand raster operations, allowing logical combinations of bitmap parts.

The CGI is a very large and complex set of functions, standardization of which is still far from complete.

Many of the functions are inappropriate for many devices, however it is not a straightforward matter to identify simple subsets of the functions and group them in sensible ways. The current draft uses the idea of a constituency profile. Such profiles define sets of functions and their precise capabilities for particular classes of CGI users. It is intended that the CGI standard itself will define a number of such profiles related to the needs of GKS, and other profiles can be standardized through the Registration procedure.

1.6. Registration

During the evolution of all these standards it has become obvious that it is not possible to standardize everything at once. In particular, there are a number of graphical elements that can be found in a bewildering number of varieties, for example marker types. Rather than delay the standards in progress by trying to get agreement on extensive lists of such elements for each standard in turn, the documents now just refer to a single registration mechanism and mandate only a very small number of such elements.

The US National Institute of Science and Technology has been approved as the Registration Authority for the Register of Graphical Items. The Procedures for Registration of Graphical Items will be published in the form of an ISO/IEC Technical Report, but this is likely to be processed as an International Standard at its first review.

1.7. Conformity Testing

The Commission of the European Communities is establishing a European Conformance Testing Service for standards in the information technology and telecommunications areas. Three European laboratories have set up a testing service for GKS: AFNOR in France, GMD in Germany and the National Computer Centre in the UK.

A standard entitled ''Conformance Testing of Graphics Standards'' is being prepared. Topics to be addressed include guidelines for conformance sections of functional standards, procedures for developing a test suite and procedures for running a test service.

1.8. Compatibility

As can be seen from the descriptions above, the degree of compatibility between the standards is not as great as one would expect. The work has tended to fragment with individual groups of experts concentrating on one or two of the many activities. It is now recognized that there is a need for a coherent reference model for computer graphics which will provide a framework for future standardization activities and a coherent programme of work.

2. Reference Model

2.1. Introduction

Serious work on a Reference Model for Computer Graphics started at the ISO meeting at Timberline in July 1985. A Task Group with membership from the GKS, GKS-3D, PHIGS, CGI, CGM and language bindings working groups came together under the chairmanship of F.R.A. Hopgood to put forward a simple model for comment.

A major problem identified early on was the relationship between the workstation interface of the functional standards and CGI. While some believed the two were synonymous, the CGI Working Group believed they had a much wider remit.

The CGM Working Group had established an International Standard based on a clean concept of a *picture* to be captured and restored. GKS, on the other hand, had much more the concept of graphical information flowing to some subset of open workstations with the arrival of information at the workstation display being determined by a number of controls. Making a clean interface between the two standards was difficult.

PHIGS, designed as a structuring facility for computer graphics, had also made changes to the primitive set and the way operator attributes, such as highlighting, were controlled at the workstation. Although PHIGS could have been designed with GKS-3D as the viewing back-end to the system, this was not the case.

In consequence, a set of graphics standards had been produced over a 10 year period with a great deal of similarity and common concepts but with minor incompatibilities due to the different times at which they were produced and the different people involved. It was clearly going to be difficult to produce a Reference Model of the existing set of standards with clean concepts. The approach had to be to define a Reference Model having a distillation of the current concepts in use and use this as the basis for the next generation of standards. GKS, the oldest of the set of standards, would be coming up for its Review in a few years time.

2.2. Strand Model

An *ad hoc* Committee on Reference Models was established and this met in Frankfurt in February 1986. A major input to that meeting was a paper by Graham Reynolds[9] emanating from the Modular Graphics Systems Project[1] at the University of East Anglia. This defined a novel process oriented graphics system architecture for emulating a variety of computer graphics systems. It was proposed that this could be used as th. basis for a Reference Model.

The underlying conceptual models of most standard graphics systems, in particular of those existing and proposed international standards for graphics, and of many existing graphics packages, are most often seen as being graphics processing pipelines. In the case of graphics output, graphics data is refined as it passes down the pipeline, by associating graphical attributes, transforming coordinates, clipping etc., until it reaches a form which is suitable for display on a particular workstation or device. Graphical input can be viewed as a pipeline of processes transforming the data resulting from some input interaction into a form suitable for use by the application. The input interaction may also involve processes from the output pipeline in order to achieve any desired prompts and echoes. Clearly, the composition of these pipelines and the order of components within them may differ widely between models, however there are often a reasonable number of components common to most. Examples of these are transformations, attributes, clipping, storage etc. These common components play an equivalent role in each model, even though the internal details of the components will most likely differ. It can be shown that a large number of the differences between graphics system models can be expressed in terms of the different orderings (or configurations) of these components.

The Reynolds abstract reference model of graphics data states developed this processing pipeline model by isolating the smallest incremental changes to the states of graphics information (or storage areas), and by defining when graphics data undergoes transitions between these states by the application of specialized processes.

The data states are grouped together to form strands of processing, where a particular strand is concerned with a subset of the overall intended graphical effects. Five major strands can be identified in most standard graphics systems, as follows:

(1) attribute strand;

(2) transformation strand;

(3) clipping strand;

(4) dimensionality strand;

(5) storage strand.

The processing strands are illustrated in figure 3, which also indicates how a specific graphics pipeline can be configured by ordering the state transitions on and between strands. The Frankfurt meeting identified a list of concepts that needed to be included in a Reference Model:

(1) pipelines;

(2) levels/interfaces;

(3) multiplexing;

(4) attribute binding;

(5) elaboration;

(6) instantiation;

(7) language bindings and encodings;

(8) data coding versus procedural interface;

(9) resource sharing;

(10) input model;

(11) application interface;

(12) operator interface;

(13) primitives, attributes;

(14) storage structures;

(15) workstations;

(16) metafiles;

(17) raster graphics.

The strand approach was looked at as an alternative to the more normal pipeline description of computer graphics.

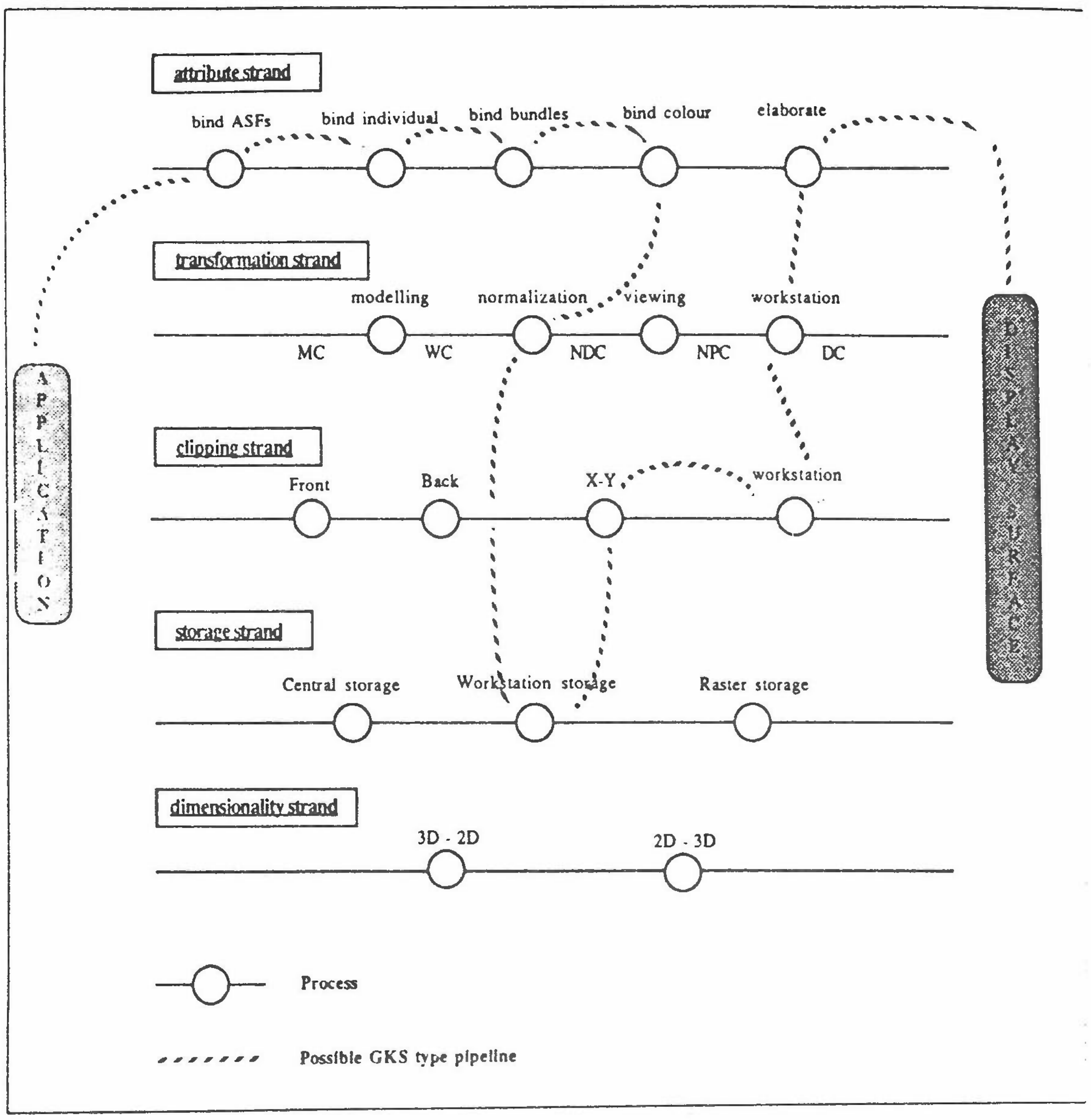


Figure 3

2.3. External Reference Model

After the Frankfurt meeting, two independent approaches were considered. The first concentrated on establishing a Reference Model that was primarily concerned with how other standards would interact with the computer graphics standards. The second concentrated on establishing an internal reference model for computer graphics showing how the various concepts in graphics should fit together.

The External Reference Model was based on the pipeline approach establishing a 7 stage pipeline for input and output. The output stages were seen as:

(1) *Conceptualization*: the mapping of the application's requirements into graphical terms.

(2) *Formulation*: the creation of graphical information.

(3) *Elaboration*: the mapping of graphical information onto the abstract picture on a workstation.

(4) *Generation*: the mapping of some part of the abstract picture onto a virtual display surface.

(5) *Realization*: the use of real attributes on the workstation to define the picture.

(6) *Production*: the process of causing the image to appear.

(7) *Visualization*: the process of inspecting the output by the operator.

A similar set of stages did the reverse process for input.

Work on the external reference model continued until January 1989 with several refinements of the document.

2.4. Components and Frameworks

The major input to the internal reference model came from BSI using the work of Arnold and Reynolds on strands and Duce on Formal Specification.[2] Out of these came a components and frameworks model for describing graphics standards.

Components can be thought of as basic concepts such as output primitives, attributes, pictures, views etc. Frameworks define how these components fit together in a particular standard. Thus a framework statement might be that pictures in this standard can only be constructed from a sequence of output primitives that have all their attributes bound to them.

As for the external reference model, the internal reference model went through a number of refinements with attempts at describing existing standards in terms of the model. As a separate activity, the relationship between components and abstract data types was established.

2.5. A Single Reference Model

A meeting was held in Paris in January 1989 to consider the two activities - external and internal. The major output from the meeting was a decision to merge the external and internal reference model into a single activity. The meeting elaborated the basic concepts or components in the Reference Model and attempted to define both the internal and external relationships. Four major concepts were agreed:

(1) *Pictures*: the current contents of a space.

(2) *Collections*: a storage structure associated with primitives and their related attributes.

(3) *Metafiles*: a mechanism for storing and retrieving pictures.

(4) *Archives*: a mechanism for storing and retrieving collections.

A Reference Model based on these 4 major concepts was seen as being both feasible and able to relate to existing standards.

A subsequent meeting in Darmstadt provided further input to the Reference Model including a desire to have more symmetry between output and input and an ability to have attributes jointly owned by the output primitive and the associated input device.

2.6. Summary

A Working Draft Reference Model has now been completed by the acting Document Editors D.A. Duce and F.R.A. Hopgood. This will be presented to the complete ISO Working Group in October 1989. Its current state is given in Appendix A.

3. Revision of GKS

3.1. Introduction

The international Review of GKS started with a Workshop sponsored by Eurographics in Disley in September 1987.[11] The Workshop split into four major activities:

(1) concepts;

(2) storage;

(3) primitives;

(4) input.

The major concerns in each area are given below.

3.2. Concepts

There was less clarity in the concepts of GKS than there ought to be. A major problem of any review would be to decide whether a slavish upward compatibility with GKS was necessary or to admit that the original concepts were not clean and that any revision would almost certainly cause some GKS programs to cease to work as they did before.

Additional primitives had been defined for the newer standards. A major question was whether these should be added to GKS or the GKS model extended to include them as a subpart.

The text primitive continued to give concern. It did not have the functionality required by the typesetting community yet was the most exotic primitive in GKS. Either it should be extended to cover the wider community or reduced to a level similar to the other primitives.

There was strong support for a more precise definition which left language binding considerations to that standard. A new GKS should not look like a FORTRAN subroutine library.

3.3. Storage

The segment model of GKS was very restrictive. Also, it did not completely specify the storage required. For example, font definitions had not been integrated into the GKS storage model. A facility for defining macros was not available.

The main conclusions were that there was a real need for a workstation independent storage system. If the GKS Workstation Independent Storage System was available, the possibilities for implementors were greatly enhanced. A new standard should make a global storage mechanism mandatory and that would simplify the overall model.

The nameset and filter mechanism of PHIGS was reviewed in some detail. The conclusion was that the segment facilities of GKS could be subsumed into a more general naming model. It would allow all the current functionality but would also allow all the advantages of data stored effectively in a relational database.

3.4. Primitives

The major concerns were the level of workstation dependencies which made it impossible for a device independent view of graphics to be available at the NDC level. Text extent was a particular problem.

The main concerns were to increase the portability and make the standard more device independent.

There was a belief that text was not really a basic primitive. CELL ARRAY could be interpreted as a special case of FILL AREA. The need for an extension to FILL AREA had to be quantified.

3.5. Input

The input model in GKS was defined at Abingdon in 1981. Not all of the input model defined there was included in GKS. In particular, there was a good description of machine independent input tools and how they could be set up.

The main conclusions were that all applications programs should have the ability to build logical input devices and be responsible for the echo and control of the device where appropriate.

3.6. Tucson

An ISO meeting at Tucson in July 1988 defined the scope and goals for the GKS Review. It also established a number of *ad hoc* Rapporteur Groups to provide input to the Review - input model, relationship to window management, text etc.

It was decided to split the work needed into a Maintenance Review of GKS and an activity aimed at producing a new Application Interface which catered for those areas not supported by GKS.

The Maintenance Review of GKS was targeted at:

(1) making any necessary editorial changes and technical corrections;

(2) enhancing portability by reducing implementation and workstation dependencies;

(3) considering new functionality coming in from the later standards;

(4) reviewing the interaction of GKS with metafiles.

3.7. Ilkley

The first GKS Review Meeting took place in Ilkley in March 1989. The BSI made a strong case for making GKS better defined even if this meant a loss of compatibility and for extending GKS to remove the major deficiencies encountered by users.

The Ilkley meeting agreed that:

(1) New functionality should be considered in making the standard closer to CGM and CGI. The naming and selection of primitives in PHIGS should be examined. An improved text model was needed.

(2) The Error Reporting mechanism needed to be less language dependent.

(3) More work should be delegated to the language bindings.

(4) The use of special workstations for long term storage and segments was queried.

(5) The rich level structure of GKS had not been used and required simplification.

(6) The compatibility with GKS (ISO 7942) should not be more restrictive than the current variation allowed in implementations.

(7) Packaging of functions in a more abstract way would lead to a more consistent document.

(8) A clear definition of an NDC picture was needed if a sensible interface to CGM was to be achieved.

(9) Primitives with their geometry completely defined at the NDC picture should be considered.

The document editors were asked to produce a revised document based on these guidelines for discussion at the ISO meeting in Brazil in October 1989. Appendix B gives the revised text to be submitted to that meeting by Brodlie, Duce and Hopgood.

4. Summary

The new Reference Model Document and the new GKS (GKS-N) are the initial drafts of the next generation of computer graphics standards. At this stage they are far from complete and will have significant changes made to them during the Review procedure. They are published here in order that a larger audience can consider the road being taken in future graphics standardization.

References

1. D. B. Arnold, G. Hall, and G. J. Reynolds, ''Proposals for Configurable Models of Graphics Systems,'' *Computer Graphics Forum* **3**(3) pp. 201-208 (1984).
2. D. B. Arnold, D. A. Duce, and G. J. Reynolds, ''An Approach to the Formal Specification of Configurable Models of Graphics Systems,'' in *Proceedings of Eurographics 87*, ed. G. Marechal, North Holland (1987).
3. D.B. Arnold and P.R. Bono, *CGM and CGI - Metafile and Interface Standards for Computer Graphics,* Springer-Verlag (1988).
4. K. W. Brodlie, L. R. Henderson, and A. M. Mumford, ''The CGM a metafile for GKS?,'' *Computer Graphics Forum* **6**(2) pp. 87-90 (1987).
5. G. Enderle, K. Kansy, and G. Pfaff, *Computer Graphics Programming, GKS - The Graphics Standard,* Springer-Verlag (1987). (Second Edition)
6. F. R. A. Hopgood, D. A. Duce, J. R. Gallop, and D. C. Sutcliffe, *Introduction to the Graphical Kernel System (GKS),* Academic Press (1986). (Second Edition)
7. W. Huebner and Z.I. Markov, ''GKS-based Graphic Programming in Prolog,'' in *GKS Theory and Practice*, ed. P.R. Bono and I. Herman, Springer-Verlag (1987).
8. ISO, ''Information processing systems - Computer graphics - Graphical Kernel System (GKS) functional description,'' ISO 7942, ISO Central Secretariat (August 1985).
9. G. J. Reynolds, ''A Token Based Graphics System,'' *Computer Graphics Forum* **5**(2) pp. 139-146 (1986).
10. M. R. Sparks and J. R. Gallop, ''Computer graphics language bindings: programmer interface standards,'' *Computer-Aided Design* **19**(8) pp. 418-424 (1987).
11. _, ''GKS Review Workshop,'' *Computer Graphics Forum* **6**(4) pp. 367-369 (1987).

Appendix A

Reference Model

Information processing systems - Computer graphics - Computer Graphics Reference Model

0 Introduction

The Computer Graphics Reference Model (CGRM) defines an architecture for computer graphics. Its purpose is to provide a consistent terminology for computer graphics and establish the relationship between the concepts which make up the reference model. It should be used in describing specific standards and in the relationship between graphics standards and the environment in which they exist.

This standard will provide the basis for the development of future computer graphics standards and ensure their long term coherence based on objective rational foundations. Existing graphics standards will not necessarily fit precisely into the Reference Model. However, experience with the current standards has significantly influenced the model.

International Standards related to computer graphics exist or are under development in the following areas:

a) Open System Interconnection - Basic Reference Model;

b) Virtual Terminal Protocols and Terminal Management;

c) File Transfer, Access and Management Protocols;

d) Office Document Architecture and Interchange;

e) Text and Office Systems;

f) Exchange of Product Model Data;

g) Character Sets and Coding;

h) Open Distributed Processing.

1 Scope and field of application

This International Standard defines a structure within which current and future International Standards for computer graphics shall be compared and their relationships described.

This International Standard does not define how computer graphics standards shall be defined and developed. It does not specify the functional descriptions of computer graphics standards, the bindings of those standards to programming languages, or the encoding of graphical information in any coding technique or interchange format. It is neither an implementation specification for systems incorporating computer graphics, nor a basis for appraising the conformance of implementations.

This International Standard, the Computer Graphics Reference Model (CGRM), defines a set of concepts and their inter-relationships which should be applicable to the complete range of future graphics standards.

CGRM defines computer graphics output in terms of output primitives which make up pictures that are output to the operator. The operator defines input values that are transmitted to the application in an appropriate form. Any connection of output generated to input received is handled by the application. To allow complex graphical images, CGRM defines a storage facility called the collection from which pictures may be composed.

This standard may be applied to:

a) Verify and refine user requirements for computer graphics;

b) Identify needs for computer graphics standards and external interfaces;

c) Refine individual models from the user requirements for computer graphics;

d) Define the architecture of new computer graphics standards;

e) Compare computer graphics standards.

CGRM provides four levels of abstraction that correspond to the application, virtual, logical and physical environments. CGRM defines the operations on pictures and collections appropriate in each environment.

2 Definitions

For the purpose of this International Standard, the following definitions apply.

2.1 application: The external entity that accesses the application environment. Applications are not modelled in the CGRM, but their interactions with computer graphics are modelled.

2.2 application environment: The environment closest to the application interface.

2.3 application interface: The interface provided by the application environment to the application. This is the only interface between the application and the application environment, and consequently the graphics system.

2.4 archive: A mechanism for representing a collection for storage, retrieval and transmission.

2.5 clipping: Restriction of the geometric shape of an output primitive to a region of interest.

2.6 collection: A named structured assembly of entities which can be transformed into a set of output or input primitives.

2.7 environment: A subdivision of CGRM at a given level of abstraction. The definition of the environment includes the definition of its primitives, picture, an (optional) set of collections and (optional) associated state information. Each environment contains at least one coordinate space as part of its definition. A set of functions is provided on the picture and collection.

2.8 geometry: The property of a primitive used to define nearest.

2.9 input primitive: The atomic unit from which input is composed by the application. There may be more than one class of input primitive. An input primitive consists of an input value, whose type depends on the class. An input primitive may have associated properties.

2.10 logical environment: The environment between the virtual and the physical environments. Output primitives contain complete geometric and rendering descriptions.

2.11 metafile: A mechanism for representing a picture for storage, retrieval and transmission.

2.12 operator: The external entity that observes the contents of the physical picture and provides physical input values. Operators are not modelled in the CGRM, but their interactions with computer graphics are modelled.

2.13 operator interface: The interface provided by the physical environment to the operator.

2.14 output primitive: The atomic unit from which graphical output is composed. There may be more than one class of output primitive. An output primitive consists of a geometric shape. An output primitive may have associated properties.

2.15 physical environment: The environment closest to the operator interface.

2.16 picture: A spatially structured set of output primitives at a given environment level.

2.17 post: An operation which transforms part of the picture at one environment level to a picture at the next lower environment level. Posting can be achieved by "posting" the collections making up the picture rather than posting the picture itself.

2.18 property: A value associated with a primitive whose meaning is dependent on the class of the primitive.

2.19 rendering: Differentiation of two primitives of the same type by means other than their geometry.

2.20 transformation: An operation that achieves a transition from one environment to another.

2.21 traversal: An operation which transforms a collection or part of a collection to produce the picture or part of the picture within the same environment.

2.22 trigger: An operation which transforms some part of the input memory at one environment level to some part of the input memory at the next higher environment level.

2.23 virtual environment: The environment between the application environment and the logical environment. Output primitives in the virtual environment contain complete geometric descriptions.

3 Conformance

The concepts and their interrelationships in CGRM provide the framework in which all future computer graphics standards should be defined.

Future standards should express their main concepts in the vocabulary of the CGRM and indicate the precise constraints defined by a specific standard.

CGRM requires that future standards define environments and interactions between environments as well as application and operator interfaces.

4 The Computer Graphics Reference Model

4.1 Introduction

The Computer Graphics Reference Model (CGRM) consists of the following major concepts:

a) environments;

b) output primitives;

c) input primitives;

d) pictures;

e) input memory;

f) posting;

g) triggering;

h) properties;

i) transformations;

j) collections;

k) metafiles;

l) archives.

4.2 Environments

A CGRM environment consists of a picture, an input memory, a set of collections and possibly associated state information, defined at a specific coordinate space. A set of functions is defined that applies to the picture, input memory and collection. There are four environments as shown in figure 1.

a) application;

b) virtual;

c) logical;

d) physical.

The four environments are always present in the description of a graphics system but some of them may be transparent or null.

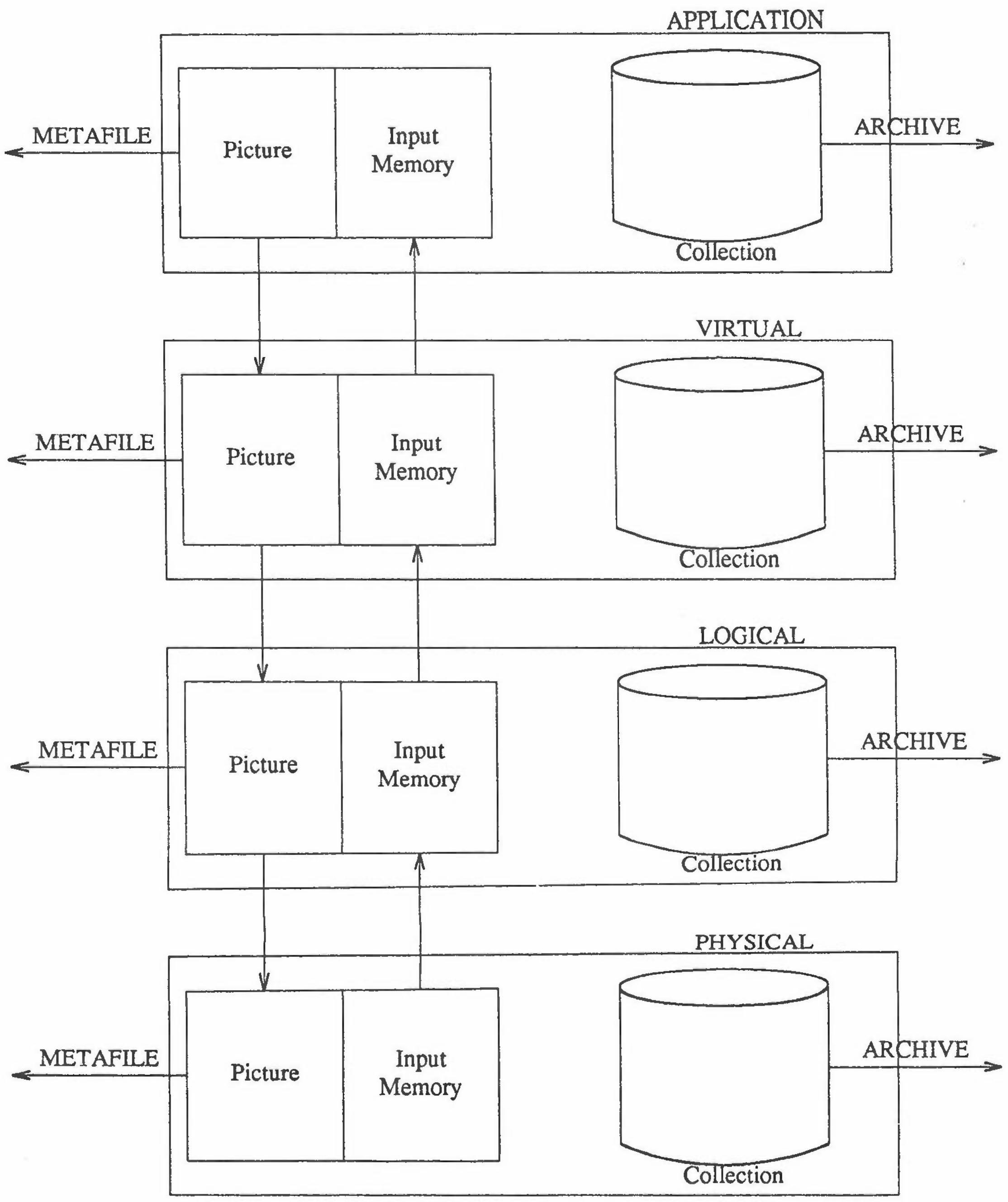


Figure 1

The main characteristics of each environment are given below.

Application environment: in this environment, output information is composed into graphics fragments with editing, composition and transformation applied. It is not necessary for the precise geometry of the picture to be defined at this stage but naming used in interaction shall be defined. Input memory is constructed in the precise form required by the application.

Virtual environment: in this environment, the graphical picture to be output is defined as a set of virtual output primitives. The geometry of these virtual primitives is completely defined so that virtual pictures are geometrically complete. All graphical devices should be capable of rendering pictures in the virtual environment. Input memory is defined in the coordinate system used in the virtual environment. Similar input primitives can only be differentiated by their associated properties.

Logical environment: associated with graphical output primitives is a set of properties associated with rendering. In the logical environment, the complete set of properties should be bound to the logical output primitive. It is possible that only a subset of output devices can precisely render the information in the logical environment. Input values are converted to device independent form with input value properties added to differentiate the origin of the input if required. The precise interpretation of the input memory by the application may not be known in this environment.

Physical environment: the environment consists of a picture in device coordinates with a specific device. Input memory will contain input values as received from the input device. It is not necessary for there to be a one-to-one correspondence between the contents of physical input memory and logical input memory. All properties associated with the input devices used will be known at this stage.

Fan-out is only allowed between the virtual and logical environments. The virtual environment may map onto more than one logical environment. A single logical environment maps onto a single physical environment as shown in figure 2.

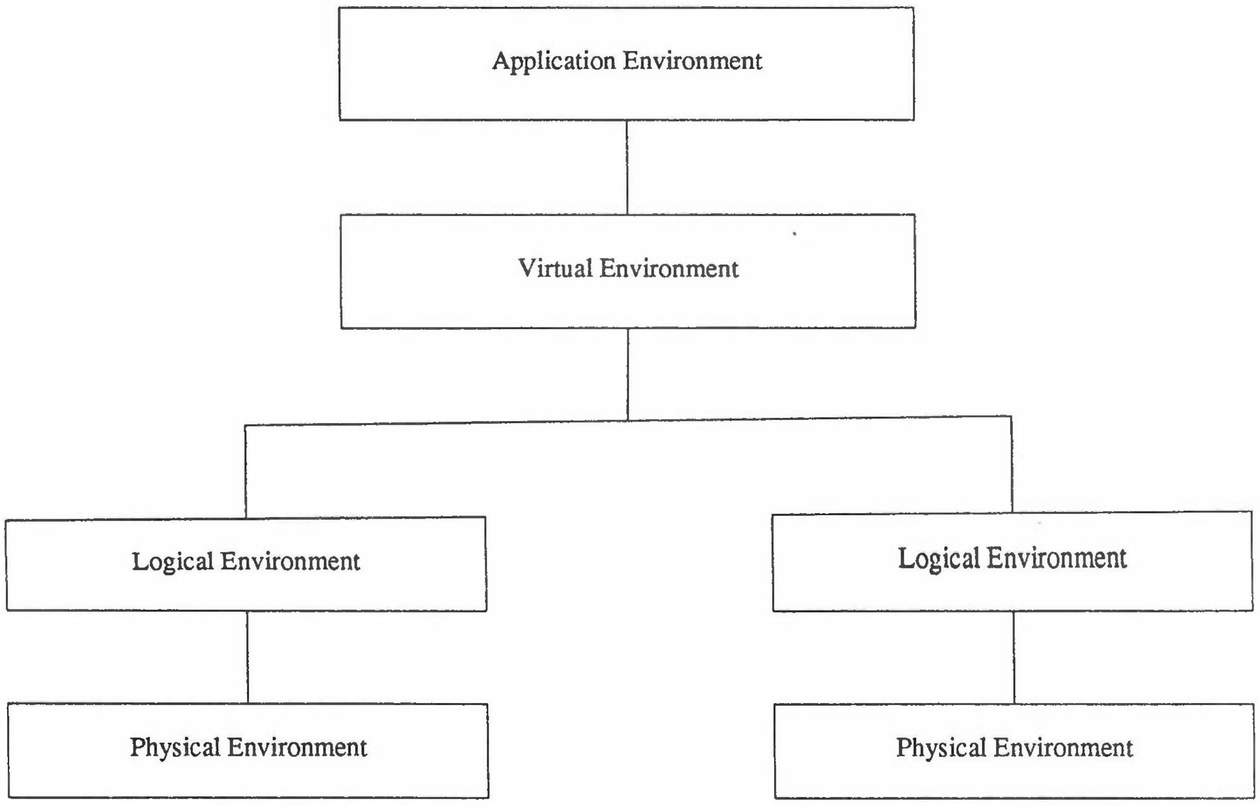


Figure 2

4.3 Output primitives

Output primitives are atomic units in terms of which graphical output in each of the four environments is described. Four types of output primitives are recognized corresponding to the four different environments. Output primitives have associated geometry and properties. Only application output primitives can be edited.

The geometry of application output primitives is not necessarily fully defined. Application output primitives can be edited, but cannot be rendered. If clipping is provided in the application environment, the results of clipping operations have to be expressible in terms of application output primitives.

The geometry of virtual primitives is completely defined. Virtual output primitives cannot be edited, clipped or rendered. Logical output primitives have both geometry and rendering completely defined.

Physical output primitives may be rendered and clipped. The result of clipping a physical output primitive is not necessarily expressible in terms of physical output primitives.

Properties of output primitives can constrain the possible input values allowed and affect the transformation applied to input values in defining input primitives at a higher environment level.

4.4 Input primitives

Input primitives are atomic units in terms of which input in each of the four environments is described. Four types of input primitives are recognised corresponding to the four different environment levels. Input primitives have associated properties and may have associated geometry. Only physical input primitives can be edited.

Physical input primitives are produced in the physical environment through the operator interface. Properties associated with physical input primitives describe the input device used.

Logical input primitives are produced in the logical environment by triggering from the physical environment. Properties associated with logical input primitives describe generic input devices.

Virtual input primitives are produced in the virtual environment by triggering from the logical environment. Properties associated with virtual input primitives include the geometry associated with the virtual picture.

Application input primitives are produced in the application environment by triggering from the virtual environment. Properties associated with application input primitives include the geometry understood by the application and any naming associated with input and output by the application.

4.5 Pictures

Graphical output is composed into pictures which can be present in each of the four environments. Specific functions exist for posting a picture at one environment to generate some part of a picture at the next lower environment level.

Pictures can be stored in metafiles. Metafiles can be retrieved and added to the current picture at this environment level, or replace it.

The following operations exist on pictures:

add output primitive
add transformed collection
copy to metafile (whole picture)
delete picture
delete output primitive
inquire property
inquire property of picture at position
post picture
retrieve picture from metafile

4.6 Input memory

Input is composed into input memory which can be present in each of the four environment levels. Specific functions exist for triggering an input memory at one environment to generate some part of input memory at the next higher environment level.

The following operations exist on input memory:

add input primitive
add transformed collection
delete input memory
delete input primitive
inquire property
trigger input memory

4.7 Posting

Posting is the operation which transforms some of the picture at one environment level to be included in the picture at the next environment level. Posting can be achieved by "posting" the collections making up the picture rather than posting the picture itself.

Posting can be automatic so that a change in the picture at one environment level automatically causes the post operation to be performed. Posting can be continuous so that the picture at one environment level is continuously transformed into the picture at the lower level. Posting can be defined to occur only when specified.

4.8 Triggering

Triggering is the operation which transforms some part of the input memory at one environment level to be included in the input memory at the next higher environment level.

Triggering can be automatic so that any change in input memory at one environment level automatically causes the triggering operation. Triggering can be continuous so that the input memory at one level is continually transformed into the input memory at the next higher level. Triggering can be defined to occur only when certain events happen.

4.9 Properties

Application properties: describe the way the operator perceives the information displayed on the device and can be used in returning information from the operator to the application. Application properties are bound to primitives in the application picture.

Virtual properties: these properties are completely defined for all virtual primitives. In particular they precisely define the geometry of the virtual picture. Virtual properties precisely define the geometry of input values in the virtual input memory.

Logical properties: these properties are defined for all logical primitives. In particular, they precisely define the rendering of the logical picture required on a device. Logical properties define generic device properties to be associated with logical input primitives.

Physical properties: these properties control what parts of the logical picture appear on the device. It is possible that the physical device will not be able to render the logical picture precisely. Input properties describe the physical input device in use.

4.10 Transformations

Transformations exist between the coordinate spaces in different environments. A post operation defines a transformation between some part of the picture at one environment level before it is added to the picture at the next lower environment level. Similarly, a trigger operation defines a transformation between some part of the input memory at one environment level before it is added to the input memory at the next higher environment level.

If the picture is composed from a set of collections, and collections exist in the two environments, a transformation of the picture from one environment to another can be realized by a transformation of collections in one environment to relevant entries in the other environment with the appropriate linking to the picture at the lower level. The post and trigger operations can change the properties associated with primitivies or entities in a collection.

4.11 Collections

A set of entities can be grouped together in a collection. Structure can exist within a collection which relates one entity with another. Collections exist at all four environments. Collections can be archived and retrieved. Collections are added to a picture or input memory by the operation 'add transformed collection'. Traversal is one method that can be used to achieve this.

Archived collections can only be retrieved at the same environment level as that at which the archive was generated. The following operations exist on collection:

copy collection to archive (whole collection)
create collection
delete collection
edit collection (only in application environment)
inquire collection
post collection
retrieve collection from archive

4.12 Relationship between output and input

There is a symmetry between output and input which is exemplified by the definition of output primitives and input primitives. The application expresses the output which the operator is to observe in terms of output primitives and the operator constructs the input on which the application is to act from input primitives.

There may be a linkage between output primitives and input primitives in that properties of output primitives, for example a transformation, may be controlled by input values and properties of input primitives, for example the range of allowable values, may be indicated by an output primitive. The linkage between input and output is described in the CGRM by shared primitives and storage of input and output at appropriate levels in the model.

Graphically represented feedback and echoes to the operator input are thus no different to other graphical output. The application may choose which picture level they appear for the first time. Similarly, the application may choose at which level the naming of output primitives or their geometry can be used by the input primitives.

Conceptually, input produces responses in the application environment. However, if echoes are defined to first appear at, for example, the logical environment and input values have a transformation which generates logical coordinate values and the echo is defined to be dependent on this value, it is permissible that an optimization exists that allows the system to take the input logical coordinate value and use it in specifying the echo without needing the application to intervene.

4.13 External relationships

The overall structure of the reference model is illustrated in figure 3.

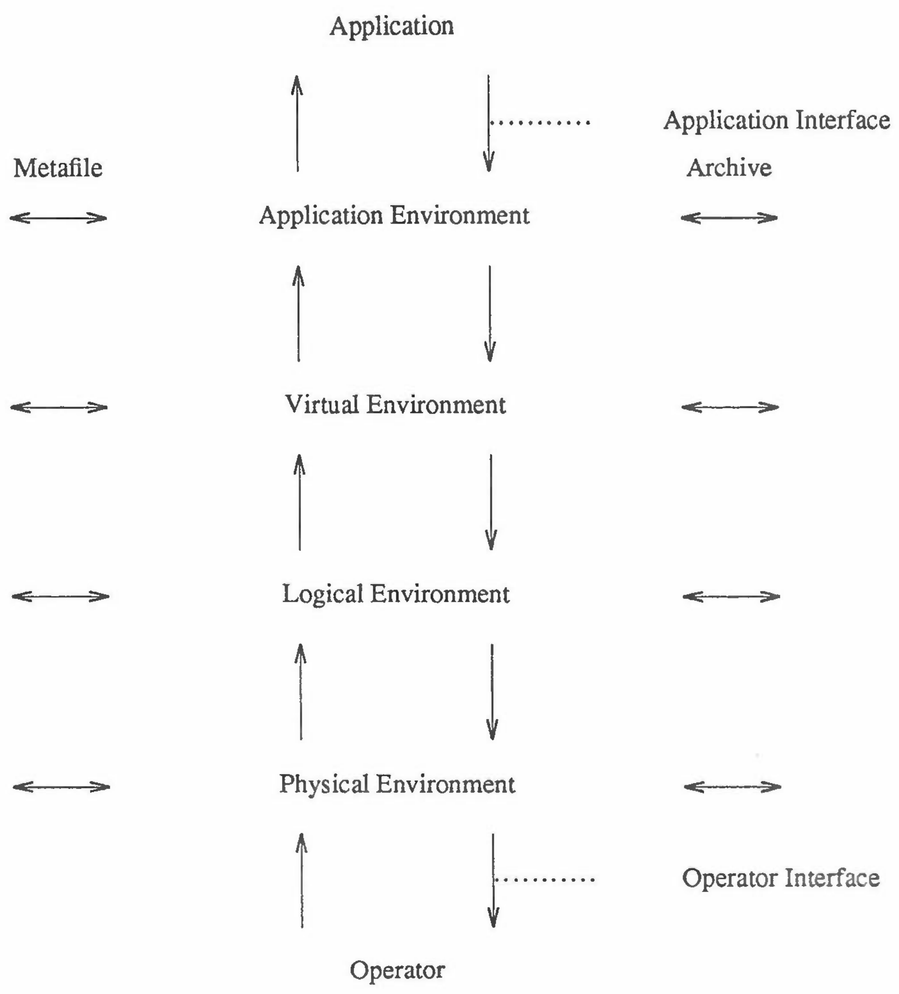


Figure 3 - Interfaces

There are two main interfaces to computer graphics:

a) *operator interface*: the interface provided by the physical environment to the operator.

b) *application interface*: the interface provided by the physical environment to the operator. This is the only interface between the application and the application environment, and consequently the graphics system.

The external entities are:

c) *operator*: the external entity that observes the contents of the physical picture and provides physical input values. The operator is not part of the computer graphics reference model.

d) *application*: the external entity that accesses the application environment. The application is not part of the computer graphics reference model.

External interfaces also exist for metafiles and archives at each environment. Internal interfaces exist between the application and virtual environments, the virtual and logical environments and the logical and physical environments.

Metafiles and archives may be generated by external agents and imported through the appropriate environment interface. Thus communication between a computer graphics system and the outside world is described in the reference model in terms of an application interface, operator interface, metafile or archive.

The four environments are always present in the description of a computer graphics system that is neither a metafile, archive, application interface, operator interface or internal interface, but it is permissible for any environment to be null, in other words to provide an identity post transformation for that environment to the next lower environment.

Annex A

Window Systems

(Non-normative Annex)

Window systems exist below the Physical Environment of the Computer Graphics Reference Model and provide a multiplexing and resource management function at this level.

The primitives of the window system are virtual bit maps with properties. The connection between the window system and the computer graphics system is shown in figure 4.

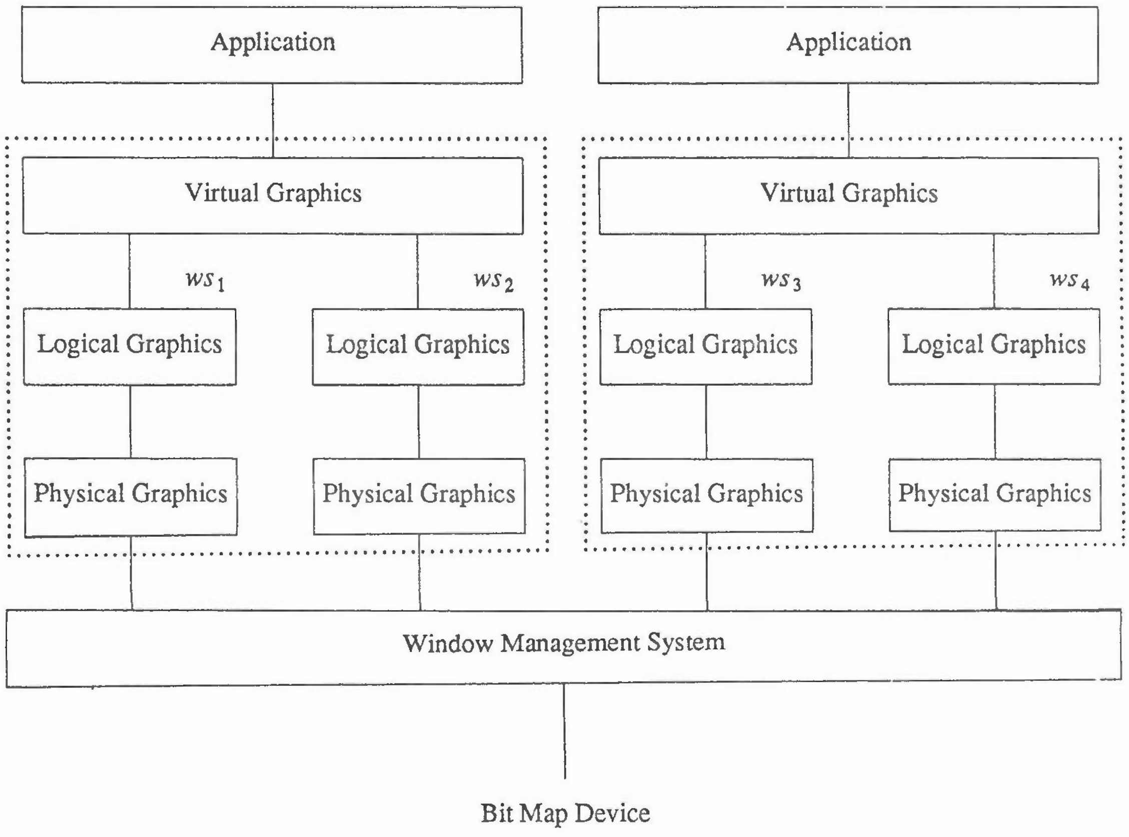


Figure 4

The window system can be elaborated using the same model as the computer graphics system, as shown in figure 5.

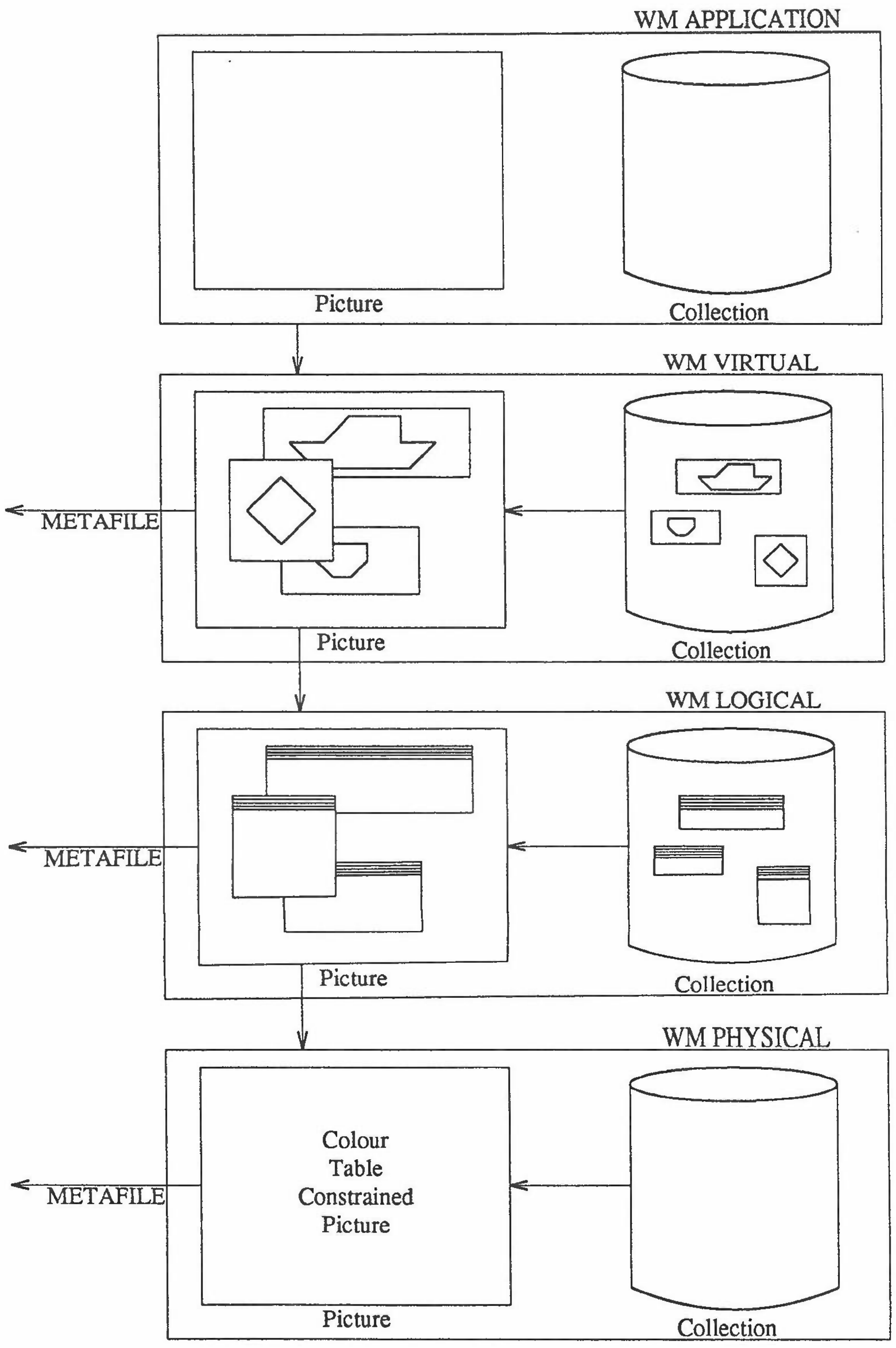


Figure 5

The window application level for some window systems at least will require a collection if the system allows window updates that change the geometry of the virtual picture. An example is a menu which displays a subset of the range of menu items. If an increase in the menu area allows more menu items to be shown, this would require the menu window to be stored at the window application level.

A collection is required at the virtual environment if the window system allows operations that do not change the geometry. Re-ordering windows on a screen could be done by the window system storing the windows for damage repair at the virtual level.

A collection is required at the logical environment if the window system does pop-up menus by storing and retrieving bit maps with the rendered information. Whether storage is done at the logical or physical environment will depend on whether changes that effect resource management cause changes to the physical picture. For example, if a window is hidden which currently uses all the colour table entries, does the uncovered window have its true colours substituted now that the colour table is not overloaded? The answer to this question determines whether the collection occurs at the logical or physical environment.

The pictures in the window environment contain primitives that are bit maps with properties. In particular, properties are used to control highlighting, visibility and detectability. For example, the focus for mouse and keyboard input is an attribute similar to visibility and detectability.

As for the CGRM, the picture is geometrically complete at the virtual environment. That is, bit maps are positioned correctly. At the logical level, borders, scroll bars, rendering of the window contents are precisely defined. At the physical environment, scarce resources such as the colour table are allocated to produce the picture to appear on the device.

There are good reasons why metafiles will be required at the virtual, logical and physical level. At the virtual environment, cut and paste operations are performed using metafile output and application processing. At the logical level, plotter output of the correctly rendered picture (without colour table resource constraints) will be achieved by metafile output. At the physical environment, metafile output is required for screen capture.

Examples of how specific window operations are performed are given below. These particular operations may be performed in other ways by some systems. In particular, nearly all operations can be performed by the application redefining bit map primitives to the window system.

Highlighting: highlighting a window is seen as an attribute of the window and is probably performed at the logical level. Highlighting would cause the allocation of colour table resources to change so would not necessarily be done at the physical environment.

Dragging: dragging a window with just a border shown can be achieved at the physical or logical level. Dragging with the complete window moving changes the geometry, and it performed no lower than the virtual environment.

Icons: the iconization of windows is done by storing icons and windows at the application environment of the window system. Whether updates of iconized pictures are retained in the application collection will depend on the window system.

Resize: as this is a geometry change, it is likely to be done using the window stored in the application collection.

Cut and paste: assuming bit maps are cut and pasted, and the system only allows the contents of bit maps to be cut and pasted, this is done by metafile output followed by input at the virtual level.

Rubber banding: similar to dragging outline box.

Stacking/reordering: changing window order is frequently done at the virtual environment. The collection contains bit maps of the windows without borders. The border information is contained in the environment state list.

Scrolling: scrolling over a limited field (for example, over a menu too large to display but of well-defined size) could be done at the application or virtual levels. For scrolling over undefined or potentially very long areas will normally require application to redefine bit map windows.

Menu selection: menus will normally be stored in the application environment collection and will cause a change in the virtual picture when popped-up, pulled-down etc.

Lens: the device that magnifies bit maps is an operation on the physical picture which may require a physical collection for picture part storage.

Colour focus: systems that give preference to a specific window in terms of colour table resource will achieve this by state information associated with the picture at the logical environment.

Radio buttons: menu devices which cause the window system to make visible or highlight some part of one of a set of windows is achieved by properties associated with the windows and possibly state information to ensure only one of several buttons is highlighted in the radio button case.

Text input: for window systems that organize text input to a window only sending text to the application on the pressing of a control key, this would be handled at the virtual environment as there is potentially some geometric changes.

Archives will depend on the requirement for persistence of information between sessions.

Collections will often contain single primitive items. However, a hierarchical set of windows is an example of a collection with a set of primitives in it if they are manipulated as a whole.

Annex B

The Relationship of Imaging to the Computer Graphics Reference Model

(Non-normative)

This annex addresses the relationship between imaging and the Computer Graphics Reference Model. This is an example of an application of the Computer Graphics Reference Model. Imaging is concerned with operations on images and extraction of features and structure from images. An image in this context is a rectangular array of picture elements. Image operations apply a basic operator over an entire region of interest of an image. Image presentation is concerned with presenting an image on a display in conjunction with graphical output, for example displaying contours on image data.

To relate the CGRM and imaging it is necessary to identify the environment or environments in which imaging operations take place. An image is identified with a collection in the application environment; image operations are then operations applied to this collection which may change the contents and structure of the collection. Transfer formats for images are identified with archives of the application collection in the CGRM. Such archives may be used to input images to an imaging system and for saving processed images.

Image presentation is a display operation on this collection in the application environment. For display purposes, the image is a particular graphics primitive and is displayed in a similar manner to other classes of graphics primitives. Displaying images with other graphical data is then treated quite naturally. The primitives representing images are refined through virtual, logical and physical environments in a manner entirely analogous to other graphics primitives.

Annex C

Existing standards and CGRM

(Non-normative)

GKS workstations correspond to the logical and physical environments. Realization of bundled aspects is done at the logical level. Transformation of coordinates from NDC to DC takes place at the posting of the logical picture. In GKS, those aspects which are definitely geometric are bound at the virtual (NDC) level. However some geometric text attributes (alignment) are not able to be completely bound until the logical level (unlike the CGRM proposal). The individual/bundled model fits into CGRM as long as the complete geometry is specified at the NDC level. This deficiency in GKS has been corrected for at least one font in PHIGS.

Segment store in GKS is identified as a virtual collection (WISS) and a logical collection (WDSS).

There are a number of areas in GKS where it does not fit precisely with CGRM. Each identifies a problemin GKS, not CGRM.

GKS-3D has more than one coordinate system involved in the posting from NDC3 to DC3 (virtual to physical). The reference model does not preclude this.

Both GKS and GKS-3D have no clear concept of a picture to be viewed. The poor compatibility with CGM identifies this. If these standards adhered to CGRM, the interaction and connection to CGM would be more precisely defined.

The structure store in PHIGS is a collection at the application level.

All three functional standards have a deferral/regeneration model which is difficult to fit into CGRM. It is believed that this is due to the imprecision of the model rather than a deficiency in CGRM.

Appendix B

GKS-N

CONTENTS

Information processing systems - Computer graphics - Computer Graphics Reference Model

0 Introduction

The Computer Graphics Reference Model (CGRM) defines an architecture for computer graphics. Its purpose is to provide a consistent terminology for computer graphics and establish the relationship between the concepts which make up the reference model. It should be used in describing specific standards and in the relationship between graphics standards and the environment in which they exist.

This standard will provide the basis for the development of future computer graphics standards and ensure their long term coherence based on objective rational foundations. Existing graphics standards will not necessarily fit precisely into the Reference Model. However, experience with the current standards has significantly influenced the model.

International Standards related to computer graphics exist or are under development in the following areas:

a) Open System Interconnection - Basic Reference Model;

b) Virtual Terminal Protocols and Terminal Management;

c) File Transfer, Access and Management Protocols;

d) Office Document Architecture and Interchange;

e) Text and Office Systems;

f) Exchange of Product Model Data;

g) Character Sets and Coding;

h) Open Distributed Processing.

1 Scope and field of application

This International Standard specifies a set of functions for computer graphics programming, the New Graphical Kernel System (GKS-N). It provides functions for two dimensional graphical output, the storage and dynamic modification of pictures, and operator input. GKS-N functions and datatypes are specified independently of programming languages.

GKS-N establishes a system for device independent graphics programming by separating picture composition and interaction from the realization of the pictures on a specific output device and the input devices used by the operator.

This International Standard is applicable to a wide range of applications that produce two dimensional pictures on vector or raster graphical devices. Operator interaction is allowed with these pictures.

2 Definitions

For the purpose of this International Standard, the following definitions apply.

2.1 application: The external entity that accesses the application environment. Applications are not modelled in the CGRM, but their interactions with computer graphics are modelled.

2.2 application environment: The environment closest to the application interface.

2.3 application interface: The interface provided by the application environment to the application. This is the only interface between the application and the application environment, and consequently the graphics system.

2.4 archive: A mechanism for representing a collection for storage, retrieval and transmission.

2.5 clipping: Restriction of the geometric shape of an output primitive to a region of interest.

2.6 collection: A named structured assembly of entities which can be transformed into a set of output or input primitives.

2.7 environment: A subdivision of CGRM at a given level of abstraction. The definition of the environment includes the definition of its primitives, picture, an (optional) set of collections and (optional) associated state information. Each environment contains at least one coordinate space as part of its definition. A set of functions is provided on the picture and collection.

2.8 geometry: The property of a primitive used to define nearest.

2.9 input primitive: The atomic unit from which input is composed by the application. There may be more than one class of input primitive. An input primitive consists of an input value, whose type depends on the class. An input primitive may have associated properties.

2.10 logical environment: The environment between the virtual and the physical environments. Output primitives contain complete geometric and rendering descriptions.

2.11 metafile: A mechanism for representing a picture for storage, retrieval and transmission.

2.12 operator: The external entity that observes the contents of the physical picture and provides physical input values. Operators are not modelled in the CGRM, but their interactions with computer graphics are modelled.

2.13 operator interface: The interface provided by the physical environment to the operator.

2.14 output primitive: The atomic unit from which graphical output is composed. There may be more than one class of output primitive. An output primitive consists of a geometric shape. An output primitive may have associated properties.

2.15 physical environment: The environment closest to the operator interface.

2.16 picture: A spatially structured set of output primitives at a given environment level.

2.17 post: An operation which transforms part of the picture at one environment level to a picture at the next lower environment level. Posting can be achieved by "posting" the collections making up the picture rather than posting the picture itself.

2.18 property: A value associated with a primitive whose meaning is dependent on the class of the primitive.

2.19 rendering: Differentiation of two primitives of the same type by means other than their geometry.

2.20 transformation: An operation that achieves a transition from one environment to another.

2.21 traversal: An operation which transforms a collection or part of a collection to produce the picture or part of the picture within the same environment.

3 Definitions

For the purposes of this International Standard, the following definitions apply.

3.1 ASF (attribute source flag): A flag for each logical attribute. If flag is set to INDIVIDUAL, the logical attribute is bound to the primitive at creation. If flag is set to BUNDLED, a bundle index is bound at creation and the logical attribute is bound at the logical level from the bundle table.

3.2 attribute: A particular property that applies to an output primitive.

3.3 baseline: A horizontal line within a character body (see figure 1) which, for many character definitions, has the appearance of being a lower limit of the character shape. A descender passes below this line. All baselines in a font are in the same position in the character bodies.

3.4 bottomline: A horizontal line at the bottom of the character body (see figure 1) which is just below all descenders in a font. All bottomlines in a font are in the same position in the character bodies.

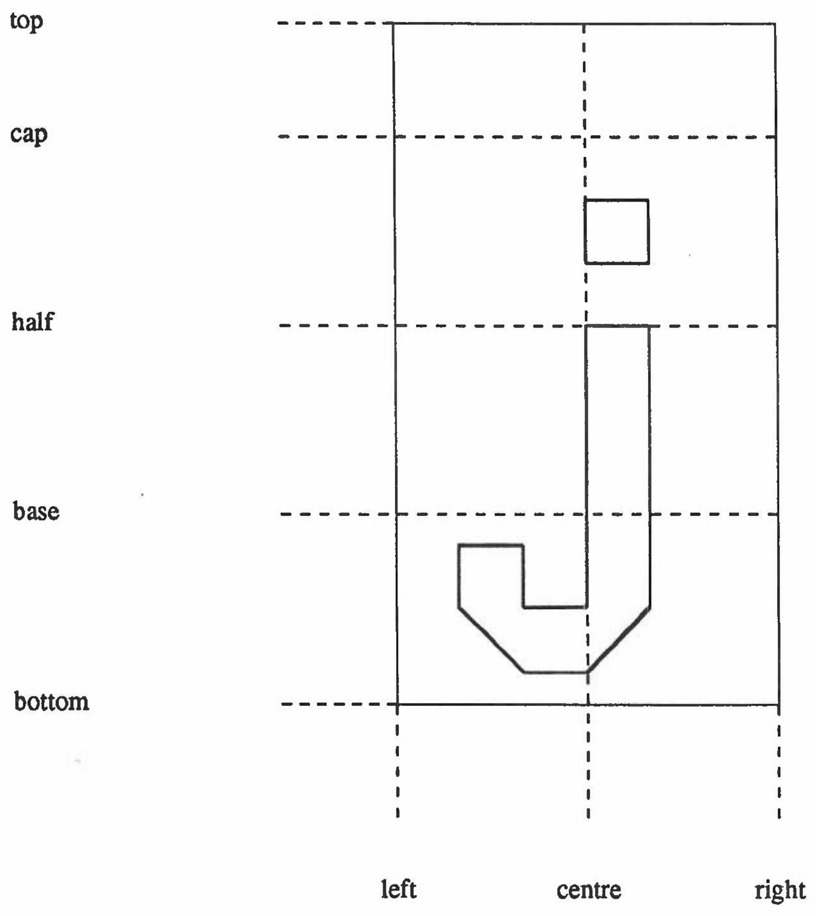


Figure 1 - Font description coordinate system

3.5 bundle index: An index into a particular bundle table.

3.6 bundle table: A workstation dependent table. A table exists for each class of bundled primitive. Each table entry defines the values of the logical attributes for that class of bundled primitive corresponding to a particular value of the bundle index. Whether these values are bound to particular bundled primitives depends on the settings of the corresponding ASFs.

3.7 bundled primitive: A type of output primitive for which significant control of appearance is allowed by workstations.

3.8 capline: A horizontal line within a character body (see figure 1) which, for many character definitions, has the appearance of being the upper limit of the character shape. An ascender may pass above this line and in some languages an additional mark (for example an accent) over the character may be defined above this line. All caplines in a font are in the same position in the character bodies.

3.9 cell array: A class of bundled primitive consisting on creation of a rectangular grid of equal size rectangular cells, each having a single colour index.

3.10 centreline: A vertical line bisecting the character body (see figure 1).

3.11 character body: A rectangle used by a font designer to define a character shape (see figure 1). All character bodies in a font have the same height.

3.12 choice device: A class of logical input device defining one of a set of alternatives.

3.13 closed path: A path where the last point is the same as the first.

3.14 colour index: An index into a particular colour table.

3.15 colour table: A workstation dependent table. Each table entry defines the colour corresponding to a particular value of the colour index.

3.16 contour: A shape which defines the outline surrounding a sub-path.

3.17 description table: A table whose entries specify the capabilities of an implementation (GKS description table) or a workstation (workstation description table).

3.18 device coordinates (DC): The coordinates used to define logical and physical pictures on a workstation.

3.19 display space: The area available for displaying images on a particular workstation.

3.20 echo: The immediate notification to the operator of the current value of an input device.

3.21 fill area: A class of bundled primitive which is a closed polygon.

3.22 Generalized Drawing Primitive (GDP): A class of bundled primitive used to address special requirements such as curve drawing.

3.23 geometric primitive: A type of output primitive where the geometry is precisely defined in the NDC picture.

3.24 halfline: A horizontal line between the capline and the baseline within the character body (see figure 1), about which a horizontal string of characters in a font would appear centrally placed in the vertical direction. All halflines in a font are in the same position in the character bodies.

3.25 identification attributes: A type of attribute used to name output primitives.

3.26 locator: A class of logical input device providing a position in world coordinates and a normalization transformation number.

3.27 logical attributes: A type of attribute which defines the appearance of a bundled primitive in the logical picture. Logical attributes may have different values for different workstations.

3.28 logical input device: An abstraction of one or more physical devices that delivers logical input values to the program. Specific logical input devices can be of class LOCATOR, STROKE, VALUATOR, CHOICE, PICK and STRING.

3.29 logical input value: A measure value delivered by a logical input device.

3.30 logical picture: the view of the NDC picture for a particular workstation in which logical attributes are bound to output primitives.

3.31 marker: A glyph with a specific appearance which is used to identify a particular position.

3.32 measure: A value which is determined by one or more physical input devices and a mapping from the values delivered by the physical devices.

3.33 nameset: An attribute in the form of a set of names associated with the output primitive.

3.34 NDC attributes: A type of attribute bound to primitives when they are created.

3.35 NDC picture: the picture in which graphical output is composed by the application program and with which graphical input devices interact.

3.36 normalization transformation: A window-to-viewport transformation that maps positions in world coordinates to normalized device coordinates.

3.37 normalization transformation number: An identification of a particular normalization transformation.

3.38 normalized device coordinates (NDC): The coordinates used to define the NDC picture and picture parts.

3.39 output primitive: A basic graphic element that can be used to construct the NDC picture and picture parts. Output primitives in GKS-N are of two types: geometric primitives and bundled primitives.

3.40 path: A sequence of sub-paths where the end of one sub-path is the start of the next.

3.41 physical picture: the picture constructed from the logical picture for a particular workstation by binding physical attributes. The physical picture is displayed on the workstation's display space.

3.42 pick: A class of logical input device providing the identification attributes attached to an output primitive.

3.43 pick identifier: An identification attribute of an output primitive.

3.44 picture part: A sequence of output primitives with associated attributes or an image.

3.45 picture part store: A collection of picture parts.

3.46 polyline: A class of bundled primitive consisting of a set of connected lines.

3.47 polymarker: A class of bundled primitive consisting of a set of positions.

3.48 prompt: An indication to the operator that a specific logical input device is available.

3.49 selection criterion: A rule for choosing elements from a sequence of output primitives. In GKS-N, selection criteria are expressed as operations on namesets.

3.50 shape: An area defined by a set of closed paths.

3.51 state list: A list whose entries specify the current values of variables relating to GKS-N as a whole (GKS state list) or to a specific workstation (workstation state list).

3.52 string: A class of logical input device providing a character string.

3.53 stroke: A class of logical input device providing a sequence of positions in world coordinates and a normalization transformation number.

3.54 sub-path: A sequence of points defining the path between two end points.

3.55 text: A class of bundled primitive consisting of a position and a character string.

3.56 topline: A horizontal line at the top of a character body (see figure 1) which is just above the upper limit of all character shapes in a font. Ascenders and accents are below the topline.

3.57 trigger: A condition which determines when a logical input value is delivered by a logical input device.

3.58 valuator: A class of logical input device providing a real number.

3.59 workstation: A display space and associated input peripherals.

3.60 workstation transformation: A workstation window-to-viewport mapping which transforms positions in NDC to device coordinates, preserving aspect ratio.

3.61 world coordinate (WC): The coordinates used by the application program to define output primitives.

4 Conformance

4.1 Specification

The set of functions known as GKS-N shall be as described in Clauses 4, 5, 6, 7, 8, 9 and 10. These functions are organized in two upward compatible levels. A GKS-N implementation shall be invalid if it lies between or outside the two defined levels. In an implementation all graphical capabilities that can be addressed by GKS-N functions shall be used only via GKS-N.

4.2 Registration

For certain parameters of the functions, GKS defines value ranges as being reserved for registration. The meanings of these values will be defined using the established procedures. These procedures do not apply to values and value ranges defined as being workstation or implementation dependent; these values and ranges are not standardized.

Information concerning the Registration Authority and its procedures may be obtained on request to the Secretary General, ISO Central Secretariat, case postale 56, CH-1211 Genève, Switzerland, quoting the number of this International Standard.

5 Concepts

5.1 NDC picture

Graphical input and output in GKS-N is defined in terms of a set of virtual input and output devices. A central concept of GKS-N is the virtual or *NDC picture* (Normalized Device Coordinate picture) where graphical output is composed and with which the operator interacts using graphical input devices (see figure 2).

The NDC picture consists of a sequence of output primitives.

5.2 Output primitives and attributes

Output primitives are abstractions of basic actions a device can perform such as drawing lines and printing character strings. Associated with output primitives are *primitive attributes* which define additional properties of the primitive. For example, the size of characters to be output could be specified by a primitive attribute.

Output primitives have three sets of attributes:

a) identification

b) NDC

c) logical

Identification attributes can be used to partition output primitives into sets for a variety of purposes. The main identification attribute is the *nameset.*

NDC attributes are bound to the output primitive on creation and describe the output primitive in the NDC picture.

Logical attributes define the rendering of the logical picture to be displayed on the workstation.

5.3 Workstations

GKS-N defines a workstation as an abstract display space and associated input peripherals. Multiple workstations can be in operation together.

A *selection criterion* based on the *namesets* of the output primitives in the NDC picture defines what subset of the NDC picture is displayed on a workstation.

For each workstation, the NDC picture is transformed into a *logical picture* on the workstation. The logical picture is defined in NDC coordinates. Output primitives in the logical picture have all logical attributes and relevant NDC attributes bound to them. Logical attributes may be bound to output primitives in the NDC picture in which case the same values are bound to the equivalent output primitive in the logical picture.

The logical picture is transformed into the *physical picture* which is realized on the workstation display space. The physical picture is obtained by applying all relevant attributes to the output primitives in the logical picture.

5.4 Coordinate systems

Output primitives are defined by the application in a *world coordinate* system. The world coordinates are specified by the application. More than one world coordinate system can be specified.

The application specifies the transformation from world coordinates to NDC. NDC is a workstation independent coordinate system.

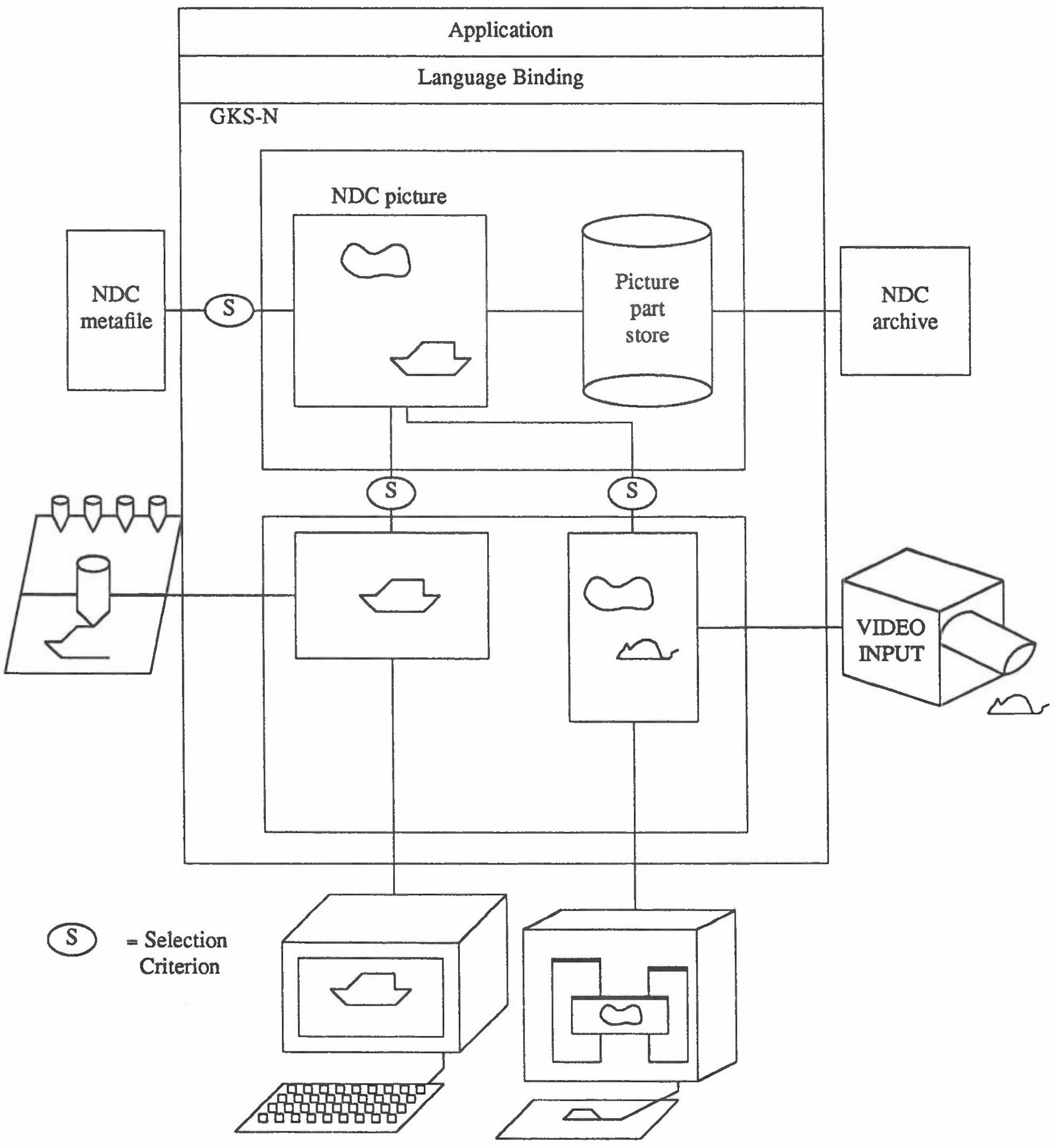


Figure 2 - GKS-N structure

The display space of a workstation has a *device coordinate* system (DC) associated with it. The logical picture is defined in NDC and the physical picture in DC. The application specifies the mapping from NDC to DC for each worksation.

Output primitives and attributes are mapped from WC to NDC by *normalization transformations* and from NDC to DC by *workstation transformations.*

5.5 Logical input devices

A *logical input device* is an abstraction of one or more physical devices. An application can define how it receives *logical input values* from logical input devices.

Logical input devices are divided into classes dependent on the datatype of the logical input value. All logical input devices of one class deliver logical input values of the same datatype.

Several operating modes are defined for every logical input device. The operating modes specify whether the operator or the application has the initiative in controlling input.

5.6 Picture part store

Sequences of output primitives with associated attributes can be defined as a *picture part* and retained in *picture part store.* Positions associated with output primitives and attributes are stored in NDC coordinates. The NDC picture can be augmented by adding the sequence of output primitives in a picture part. By applying different transformations to the output primitives in a picture part, several instances of a picture part can appear in the NDC picture.

Some predefined picture parts exist in GKS-N which are used in the definition of output primitives.

5.7 Shape store

One type of output primitive in GKS-N consists of a shape through which a picture part is extruded. The shape store contains both pre-defined shapes and shapes defined by the application. An example of a pre-defined shape is a character form. The picture part store contains some pre-defined picture parts - in particular a solid colour area covering the whole of NDC space. This picture part is used in the definition of solid colour characters and areas. Predefined picture parts are available that correspond to the colour index values available.

5.8 State lists

State lists in GKS-N precisely describe the current state of an application in its use of graphics. Two types of state list exist in GKS-N. The *GKS state list* describes the current state of the NDC picture, picture part store and the associated logical input devices. The *workstation state list* describes the current state of a workstation. Several workstation state lists can be in use at the same time.

5.9 Description tables

The details of a particular GKS-N implementation are defined in a set of description tables. The *GKS description table* contains information about the specific implementation of GKS-N. The *workstation description table* defines the characteristics of a type of workstation.

5.10 Metafiles and Archives

The contents of the NDC picture and physical picture can be stored and retrieved from metafiles. Metafiles at the physical level can be used for hardcopy output and video input.

The picture part store can be archived.

6 The New Graphical Kernel System

6.1 Initialization

To activate GKS-N, the function OPEN GKS is invoked. This defines a set of data structures (called *state lists* and *description tables*) which define the characteristics of the implementation of GKS-N in use. State lists are dynamic and may change during program execution. Description tables are static and define the characteristics of a particular system. The following data structures are initialized on GKS-N being opened:

a) Operating state list: defines the state of GKS-N.

b) GKS description table: gives information about the workstations available, size of picture part store and number of normalization transformations allowed.

c) GKS state list: provides information about the state of GKS-N as the execution of a program progresses. On OPEN GKS being invoked, it is set up with predefined default values.

d) Workstation description table: describes the characteristics of a class of workstations. One is provided for each class in the GKS-N implementation accessed.

Inquiry functions are provided to access all the information in the data structures that define the implementation and current state of GKS-N in use.

6.2 Graphical output

6.2.1 Output primitives

The graphical information that is generated by GKS-N to produce the NDC picture or picture parts in the picture part store is a sequence of output primitives.

Output primitives in GKS-N are of two types:

a) Geometric primitives

b) Bundled primitives.

Relevant primitive attributes are bound to an output primitive when it is created. The SET PRIMITIVE ATTRIBUTE function specifies the current value of an attribute in the GKS state list. The binding of attributes for bundled primitives is described in 6.2.9.

All output primitives have the following two classes of identification attributes associated with them:

c) NAMESET

d) PICK IDENTIFIER

Output primitives that make up the current NDC picture can have their NAMESET attributes changed by invoking one of the functions ADD NAME TO NDC PICTURE or REMOVE NAME FROM NDC PICTURE.

The values of all other attributes cannot be changed once they have been bound to an output primitive. However, attributes can be discarded in the transformation from NDC to logical picture if they are no longer relevant.

6.2.2 Geometric primitives

Geometric primitives are defined by a shape positioned in NDC space and a picture part (also positioned in NDC space) which is extruded through the shape to produce a patterned area in the NDC picture of the picture part being created.

Shapes can be as simple as a square area or as complex as the outlines of a text string. The picture part extruded through the shape can be as simple as an infinite plane of a single colour or as complex as a fully coloured image.

Line drawings are defined in terms of shapes which form the contour of the line. A specific function is defined for creating a shape associated with the contour of a line.

Shapes are defined in world coordinates which are transformed to NDC coordinates. The creation of geometric primitives takes place in the NDC space.

6.2.3 Shape store

All shapes that are created are stored in the *shape store*. A GKS-N implementation will have a set of predefined shapes in the shape store available to the application. These include the character fonts, markers and other useful glyphs.

Shapes are defined in three ways:

a) *Set of Closed Paths*: functions are proviced to define a shape consisting of a set of closed paths.

b) *Path Contour*: a function is provided to create a shape which is the contour of a sub-path such as a sequence of line segments.

c) *Concatenation*: a shape can be created out of a sequence of already created shapes by specifying the relative positioning of the original shapes in defining the new shape.

Shapes can be deleted from shape store when they are no longer required.

6.2.4 Shape as a set of closed paths

A shape can be defined as a set of closed paths between the invocation of the function BEGIN SHAPE and the invocation of the function END SHAPE.

A *closed path* is defined by the sequence of sub-path definitions between the invocation of the function BEGIN CLOSED PATH and the invocation of the function END CLOSED PATH.

Paths are defined using the following four sub-paths:

a) *LINES*: GKS-N defines a sub-path as a connected set of straight lines defined by a point sequence.

b) *CURVES*: GKS-N defines a sub-path as a Bezier cubic curve by specifying the start point, two control points and end point.

c) *ARC*: GKS-N defines a sub-path as an arc of a circle by specifying the start point, tangent point, end point and radius. A straight line is drawn from the start point to arc of circle of radius required followed by straight line. The arc is tangential to the two lines from the start point to the tangential point and from the tangential point to the end point.

d) *ELLIPTICAL ARC*: GKS-N defines a sub-path as an arc of an ellipse by specifying a start point, tangential point and end point. An elliptic arc is drawn from start point to end point that is tangential to the two lines from start point to tangential point and from tangential point to end point.

e) *NUB CURVE*: GKS-N defines a sub-path as a Non Uniform B-spline curve. The spline order, knots and control points are provided as parameters. A range specifies the extent of the curve.

If the end point of a sub-path is not equal to the start point of the next sub-path in a closed path definition, a single line sub-path is added to link the two points. If the start point of the first sub-path is not equal to the end point of the last sub-path, a single line sub-path is added to link the two points.

6.2.5 Inside rule

The inside of a shape is defined by the set of closed paths defining the shape and an associated *inside rule*. Two inside rules are defined in GKS-N:

a) Even-odd rule;

b) Non-zero winding number rule.

The *even-odd rule* determines whether a point is inside the shape by creating a straight line starting at the point and going to infinity in any direction. If the number of intersections between the straight line and the sub-paths making up the shape is odd, the point is inside the shape; otherwise it is outside the shape.

The non-zero winding number rule determines whether a point is inside the shape by creating a straight line starting at the point and going to infinity in any direction. Each sub-path has a direction associated with it. In the case of the LINES sub-path, it is assumed that the sub-path goes from the start point to the end point. The other sub-paths have a precise definition of direction also. Starting with a count of zero, for each intersection between the straight line to infinity and a sub-path, one is added to the count for each intersection where the line crosses the sub-path when the sub-path is going from left to right. One is subtracted from the count if the sub-path is going from right to left. If the resulting count is zero, the point is outside the shape; otherwise it is inside.

For a simple shape consisting of a set of non-intersecting closed paths none of which intersect themselves, the two inside rules have the same definition of inside and outside.

Only the points inside the shape are affected by the extruded picture part in the geometric primitive.

6.2.6 Path contours

A function CREATE CONTOUR defines a shape as the contour surrounding a sub-path. The precise form of the shape created depends on the following attributes associated with the sub-paths:

a) *STYLE*: defines the style of the shape created. For example, DASHED would generate a shape consisting of a set of equal length sub-shapes interspersed with gaps of the same length.

b) *WIDTH*: defines the width of the shape created. The width can be specified using the current X or Y world coordinate or NDC. In each case, it defines a width in NDC space by the appropriate conversion.

c) *CAP*: defines the form of the two ends of the sub-path assuming it is not closed. Butted, rounded or square ends are defined.

d) *JOIN*: for sub-paths consisting of a sequence of lines this attribute specifies the join between line segments. Mitred, round or bevel joins are defined. For mitred joins, a mitre limit is defined which limits the spikes for two lines with a small angle between them.

6.2.7 Shape attributes

Each shape can have a set of shape attributes associated with it. These attach specific names to points or coordinate values within the shape. The attributes for a shape are defined explicitly by invoking the SET SHAPE ATTRIBUTE function as part of the shape creation. The following attribute names are predefined:

a) TOPY, CAPY, HALFY, BASEY, BOTTOMY, CENTREY

b) LEFTX, RIGHTX, CENTREX

c) CENTRE, ORIGIN

d) TOP, BOTTOM, LEFT, RIGHT

e) TOPLEFT, TOPRIGHT, BOTTOMLEFT, BOTTOMRIGHT

f) START, MIDDLE, END

The names in sets a) and b) define coordinate values. The names in sets c), d) and e) define positions in NDC space. The names in set f) only apply to path contours and define positions in NDC space.

6.2.8 Shape concatenation

A shape may be defined as a sequence of existing shapes with an associated *concatenation criterion*. The concatenation criterion defines the position of the second shape to the first, the third shape to the second and so on. If P is a shape attribute name of the first shape and Q is a shape attribute name of the second shape, a possible concatenation criterion might be Q *at* P.

For example, a text string output from left to right would be defined as a sequence of predefined character shapes with the concatenation criterion LEFT of NEXT at RIGHT of CURRENT.

6.2.9 Bundled primitives

Bundled primitives are divided into the six classes:

a) POLYLINE: GKS-N generates a set of connected lines defined by a point sequence.

b) POLYMARKER: GKS-N generates symbols of one type centred at given positions.

c) TEXT: GKS-N generates a character string at a given position.

d) FILL AREA: GKS-N generates a polygonal area which may be hollow or filled with a uniform colour, a pattern, or a hatch style.

e) CELL ARRAY: GKS-N generates an array of cells with individual colours.

f) GENERALIZED DRAWING PRIMITIVE(GDP): GKS-N addresses special geometrical output capabilities of a workstation such as the drawing of spline curves, circular arcs, and elliptic arcs. The objects are characterized by an identifier, a set of points and additional data. GKS-N applies all transformations to the points but leaves the interpretation to the workstation.

The interior of a FILL AREA primitive is defined in the following way (see figure 3). For a given point, create a straight line starting at that point and going to infinity. If the number of intersections between the straight line and the polygon is odd, the point is within the polygon; otherwise it is outside. If the straight line passes a polygon vertex tangentially, the intersection count is not affected. If a point is within the polygon, it is included in the area to be filled, subject to clipping. This is equivalent to the even-odd inside rule for shapes.

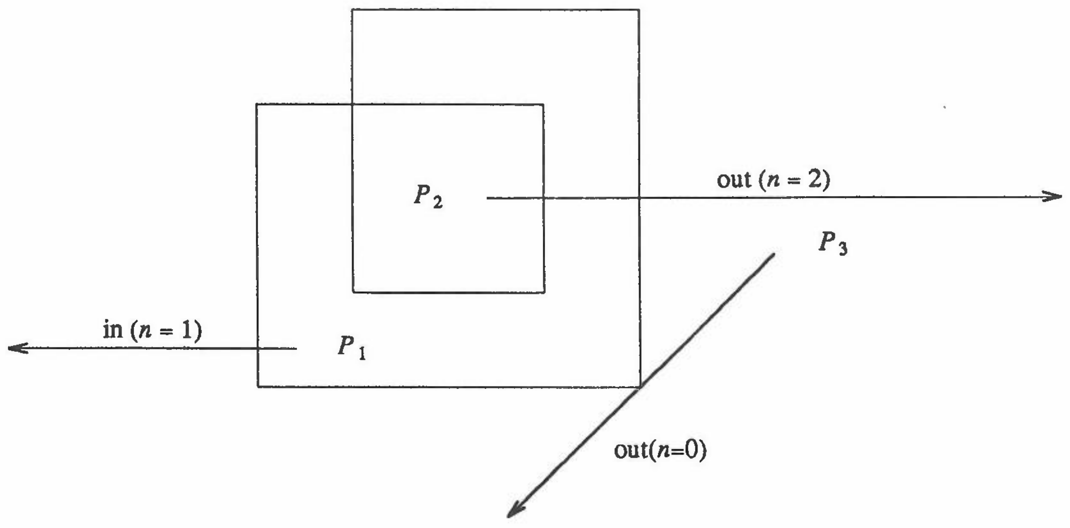


Figure 3 - Area inside a fill area boundary

When a FILL AREA primitive is clipped, the resulting new boundaries become part of the area boundaries (see figure 4). Multiple subareas may be generated.

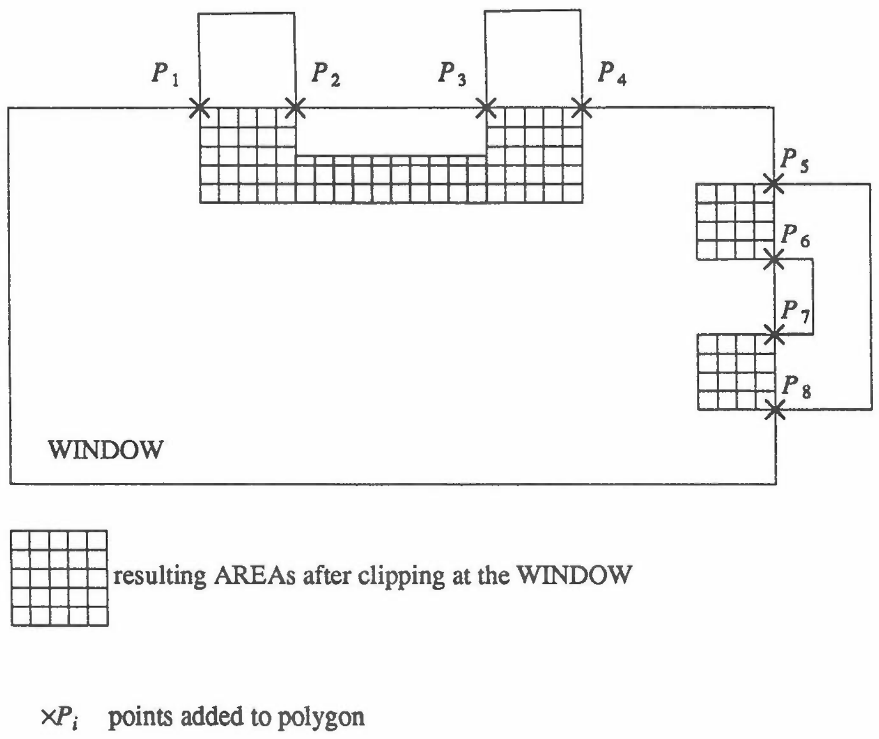

P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
WINDOW
resulting AREAs after clipping at the WINDOW
$\times P_i$ points added to polygon

Figure 4 - Examples of FILL AREA clipping

A CELL ARRAY primitive is specified by a pair of points P, Q and an array of colour indices. The points P and Q define a rectangle aligned with the world coordinate axes which is divided into a grid of DX×DY cells, where DX and DY are the dimensions of the colour index array. The colour index array is oriented with respect to the rectangle as shown in figure 5. The grid is subject to all transformations, potentially transforming the rectangular cells into parallelograms. The rules for mapping the transformed cells onto the pixels of a raster display are stated in 8.6.

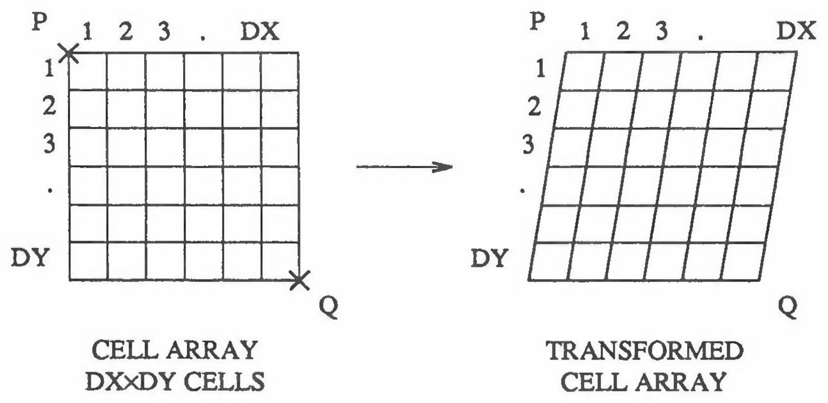


Figure 5 - Mapping of CELL ARRAYs

NDC attributes associated with the bundled primitives are:

POLYLINE	POLYLINE INDEX
POLYMARKER	POLYMARKER INDEX
TEXT	TEXT INDEX
	CHARACTER HEIGHT
	CHARACTER UP VECTOR
	TEXT PATH
	TEXT ALIGNMENT
FILL AREA	FILL AREA INDEX
	PATTERN REFERENCE POINT
	PATTERN SIZE
CELL ARRAY	none
GENERALIZED DRAWING PRIMITIVE	no explicit NDC attributes but can use attributes of other bundled primitive classes

The bundled primitive INDEX is an index into a bundle table (stored in the workstation state list) which exists on each workstation. The values in a particular bundle (or entry in the bundle table) may be different for different workstations. They control how the NDC picture is transformed to produce the logical picture. A full description of logical attributes is given in Clause 8.

Bundled primitives have both NDC and logical attributes. Current values of NDC attributes defined in world coordinates are stored in the GKS state list. When the NDC attributes are bound to their respective primitives, the values are subject to the same transformations as the geometric data contained in the definition of the primitive. Hence, current values are unaffected by changes in the normalization and workstation transformations. The difference in attribute binding for geometric and bundled primitives is illustrated in figure 6.

Logical attributes can be bound to a bundled primitive either when the primitive is created or when the primitive is added to the logical picture. The GKS state list contains a set of ATTRIBUTE SOURCE FLAGS (ASFs) which determine which attributes are bound in the NDC picture and which are postponed to the logical picture. This is described in 8.6.

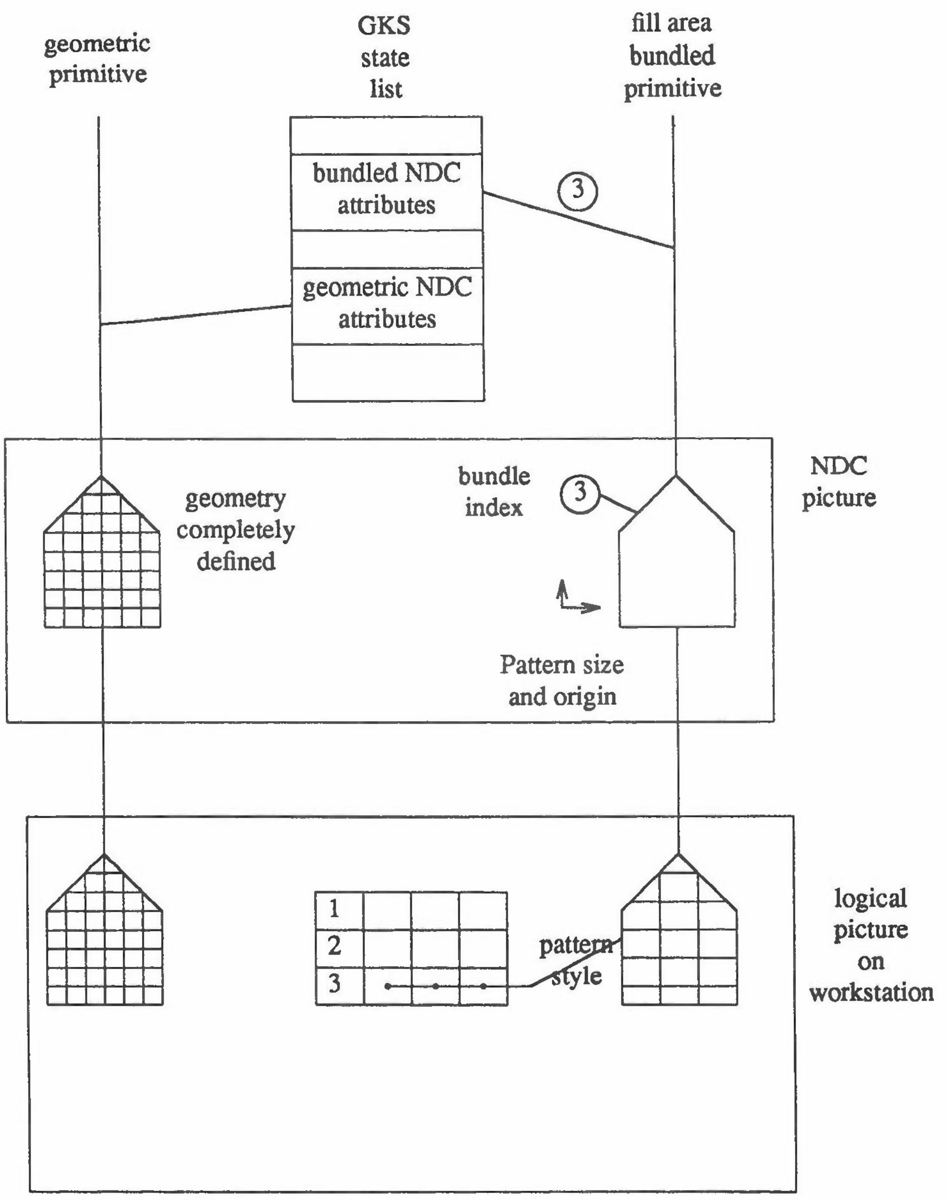


Figure 6 - Attribute binding for primitives

6.2.10 Text attributes

CHARACTER HEIGHT specifies the nominal height of a capital letter character. The CHARACTER UP VECTOR gives the up direction of a character. TEXT PATH has the possible values RIGHT, LEFT, UP and DOWN. It specifies the writing direction of the text string. For RIGHT, the text string is written in the direction of the baseline of a character implicitly specified by the CHARACTER UP VECTOR. For LEFT, the baseline direction is the opposite direction to RIGHT. For UP, the character path coincides with the direction of the CHARACTER UP VECTOR. For DOWN, it is the opposite direction to the CHARACTER UP VECTOR. For the UP and DOWN text path directions the characters are arranged so that the centres of the character bodies are on a straight line in the direction of the CHARACTER UP VECTOR (see figure 7).

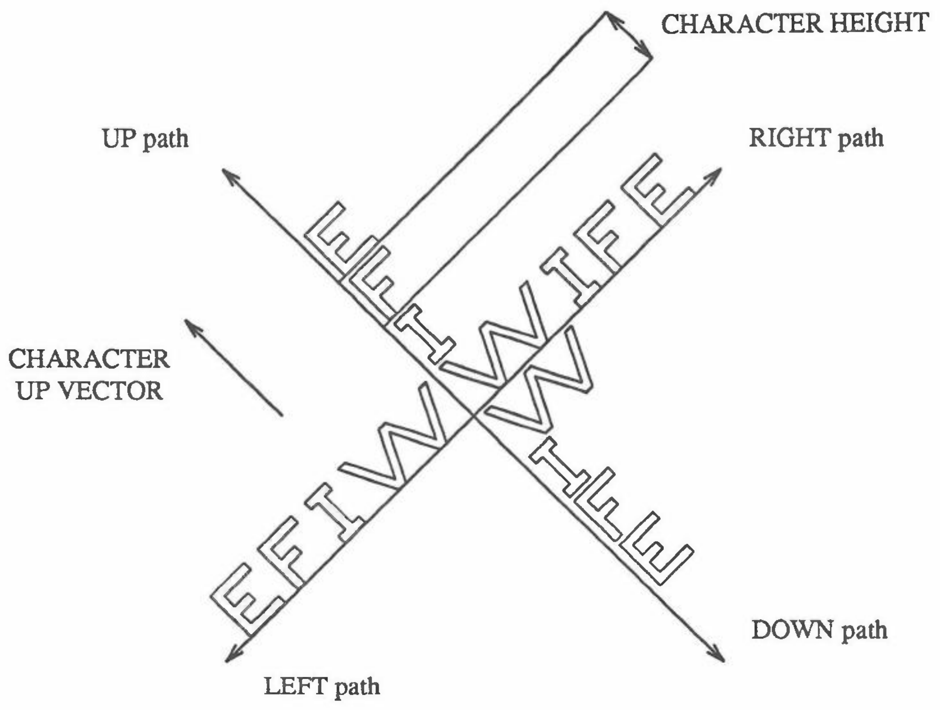


Figure 7 - Text geometric attributes

TEXT ALIGNMENT, although an NDC attribute, is more appropriately described in 8.7.4.

The initial values of the text attributes are:

CHARACTER HEIGHT	WC	0.01 (i.e. 1% of the height of the default window)
CHARACTER UP VECTOR	WC	(0,1)
TEXT PATH		RIGHT
TEXT ALIGNMENT		(NORMAL, NORMAL)

The alignment settings in figure 7 are not the NORMAL settings for text path UP and DOWN (see 8.7.4).

6.2.11 Fill area attributes

The FILL AREA primitive has the NDC attributes PATTERN REFERENCE POINT and PATTERN SIZE. If the fill area is to be rendered by a pattern in the logical picture on a workstation, the origin of the pattern and its size are the same on each workstation and are defined by these two attributes. A full description of rendering fill areas is given in 8.7.5.

6.3 Normalization transformations

In GKS-N, the application programmer can compose his graphical picture from separate entities each of which, conceptually, is defined with its own world coordinate system (WC). The relative positioning of the separate entities is defined by having a single normalized device coordinate space (NDC) onto which all the defined world coordinate systems are mapped. A set of normalization transformations define the mappings from the world coordinate systems onto the single normalized device coordinate space, which can be regarded as a workstation independent abstract viewing space. On creation, output primitives have world coordinate positions transformed to NDC coordinates before being added to the NDC picture or picture part store. Shapes are stored in NDC coordinates. A single normalization transformation is current at any one time and this is used for the transformation.

A normalization transformation is specified by defining the limits of an area in the world coordinate system (window) which is to be mapped onto a specified area of the normalized device coordinate space (viewport). Window and viewport limits specify rectangles parallel to the coordinate axes in WC and NDC (see figure 8). The rectangles include their boundaries. The normalization transformation performs a mapping from WC onto NDC that includes translation and differential scaling with positive scale factors for the two axes.

Although NDC space conceptually extends to infinity, the region of NDC space in which the viewport needs to be located and that can be viewed at a workstation is the closed range [0,1]×[0,1]. In addition, an implementation may support only a restricted range of NDCs. However, this range is always sufficiently greater than the [0,1]×[0,1] square. In particular, NDCs in the range [-7,7]×[-7,7] are always handled.

Each normalization transformation is identified by a transformation number which is an integer between 0 and 31 inclusive. The normalization transformation with transformation number 0 is the unity transformation which maps [0,1]×[0,1] in world coordinates to [0,1]×[0,1] in normalized device coordinates. It cannot be changed.

Initially, all other normalization transformations are set to a default transformation which is the same as transformation number 0. Different transformations can be specified at any time when GKS-N is open. Since GKS-N provides a number of different normalization transformations, it is possible for the application program to specify them prior to outputting the graphical picture. The separate entities in the picture are output by selecting a particular normalization transformation before outputting the associated graphical primitives. However, specifying a normalization transformation, while the graphical output is taking place, is allowed.

A normalization transformation may be selected by SELECT NORMALIZATION TRANSFORMATION, and it will be used for all output until another is selected. By default, normalization transformation 0 is selected.

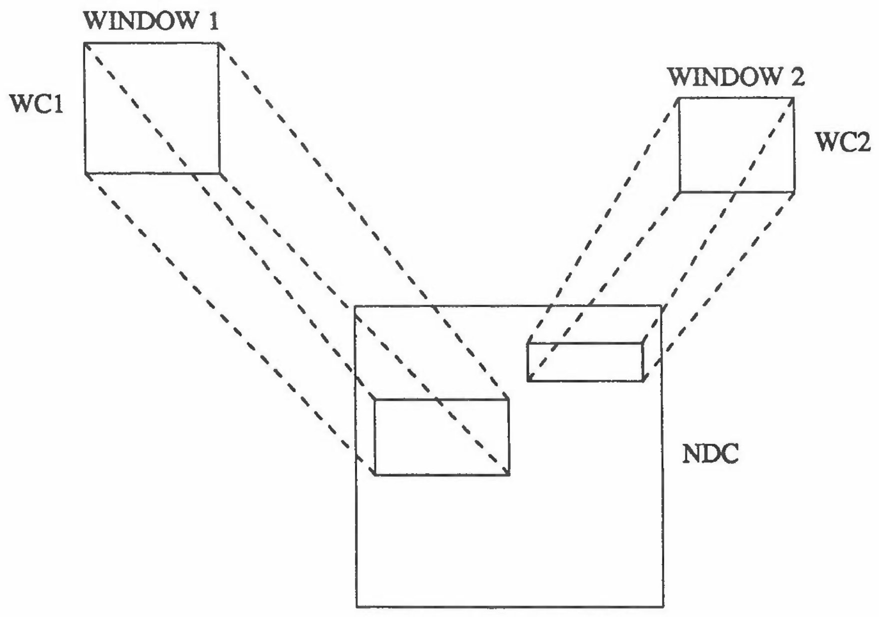


Figure 8 - Normalization transformations

6.4 Picture part store

6.4.1 Picture part creation

In GKS-N, output primitives on creation will either be added to the NDC picture or a picture part. When GKS-N is opened, primitives on creation are added to the NDC picture. When a BEGIN PICTURE PART function is invoked, primitives stop being added to the NDC picture and are added to the picture part just started instead. When END PICTURE PART is invoked, the sequence of primitives that make up the picture part are added to the picture part store (PPS) and subsequent primitives revert to being added to the NDC picture. The picture part is identified by a unique, application specified picture part name. Primitives in the picture part have attributes bound to them in the same way as primitives added to the NDC picture.

6.4.2 Picture part functions

The following functions are provided to manipulate picture parts:

a) RENAME PICTURE PART: changes the name of the picture part specified to a new name.

b) DELETE PICTURE PART: removes a picture part from the picture part store.

c) BEGIN PICTURE PART AGAIN: reopens a picture part. Output primitives created are added in sequence to the end of the specified picture part until END PICTURE PART is invoked.

d) APPEND PICTURE PART: adds one picture part to the end of the other picture part.

6.4.3 Adding picture parts to the NDC picture

A picture part can be added to the NDC picture by invoking COPY PICTURE PART TO NDC PICTURE. In sequence, each output primitive in the picture part is transformed by the specified transformation matrix and the specified NAMESET is added to the NAMESET already associated with the output primitive. This allows several instances of a picture part to appear in the NDC picture in different positions and these can be differentiated by their different NAMESETs.

6.4.4 Picture parts as images

Geometric primitives can consist of a coloured image defined as a picture part extruded through a shape. A special function is provided in GKS-N to allow flexible definition of images. The function CREATE IMAGE PICTURE PART defines a function associated with the picture part which returns a colour index given a point in NDC space.

6.5 The NDC picture

6.5.1 Introduction

The NDC picture at any time consists of a sequence of output primitives which have been added to the picture in one of three ways:

a) on creation outside the definition of a picture part;

b) by copying from a picture part;

c) on input from an NDC metafile.

The order of appearance of output primitives in the NDC picture is important in that if two output primitives overlap, the second primitive in the sequence will obscure some part of the first.

Each workstation that is open is responsible for selecting the subset of the NDC picture to be viewed and rendering it.

Associated with each output primitive in the NDC picture is a set of attributes and the clipping set and shield set in effect when the primitive was added to the NDC picture.

6.5.2 NDC picture operations

The following operations can be applied to the NDC picture:

a) DELETE PRIMITIVES: primitives whose NAMESET attribute in the NDC picture satisfies the selection criterion are deleted from the NDC picture.

b) REMOVE NAME FROM NDC PICTURE: the specified name is removed from the NAMESET of all primitives in the NDC picture.

c) ADD NAME TO NDC PICTURE: the specified name is added to the NAMESETs of all primitives in the NDC picture which satisfy the specified selection criterion.

6.5.3 NDC metafiles

It is useful to be able to capture the contents of the NDC picture and store it away for future use or transmission to another system. The function COPY NDC PICTURE TO NDC METAFILE will store the NDC picture as a picture on the specified metafile. The picture can be recovered at a later time and added to the current NDC picture by invoking COPY NDC METAFILE PICTURE TO NDC PICTURE.

6.6 Selection criterion

Functions in GKS-N select subsets of the current NDC picture according to a *selection criterion* based on the namesets of the sequence of output primitives that define the NDC picture. A function of this type is ADD NAME TO NDC PICTURE. The selection criterion is based on the Structured Query Language (SQL). The comparison operations allowed are:

a) CONTAINS

b) DOES NOT CONTAIN

c) IS IN

d) IS NOT IN

e) EQUALS

f) NOT EQUALS

Each operation specifies a nameset which is compared with the nameset of each output primitive in the NDC picture. The result is a sequence of output primitives which can be anything from the empty sequence to the complete sequence of output primitives in the NDC picture. For example, CONTAINS (RED,GREEN) would select all output primitives having namesets containing both names RED and GREEN.

More than one comparison operation can be included in the selection criterion by using the following logical operations:

g) AND

h) OR

i) NOT

For example, CONTAINS (RED,GREEN) OR CONTAINS (BLUE,YELLOW) will select those primitives with namesets containing either RED and GREEN or BLUE and YELLOW.

The selection can depend on two sub-sequences merged together by the operations:

j) UNION

k) INTERSECTION

l) MINUS

For example, CONTAINS (RED,GREEN) OR CONTAINS(BLUE,YELLOW) MINUS CONTAINS (PURPLE) will select all the primitives with namesets containing either RED and GREEN or BLUE and YELLOW except for those primitives that also contain PURPLE in the nameset.

6.7 Graphical input

6.7.1 Introduction to logical input devices

An application program obtains graphical input from an operator by controlling the activity of one or more logical input devices, which deliver logical input values to the program.

The *device identifier* defines the class of logical input device and how it relates to physical devices on a workstation.

The logical input device class determines the type of logical input value delivered. The classes and the logical input values they provide are:

a)	LOCATOR:	a position in world coordinates and a normalization transformation number.
b)	STROKE:	a sequence of points in world coordinates and a normalization transformation number.
c)	VALUATOR:	a real number.
d)	CHOICE:	a CHOICE status and a non-negative integer which represents a selection from a number of choices.
e)	PICK:	a PICK status, a nameset and a pick identifier.
f)	STRING:	a character string.
f)	COMPOSITE:	an application defined logical input device which delivers values that are composites of values of the above types.

A workstation contains at least one logical input device.

Each logical input device can be operated in three modes, called *operating modes*. At any time a logical input device is in one, and only one, of the modes set by the invocation of the function SET LOGICAL INPUT DEVICE MODE. The three operating modes are REQUEST, SAMPLE and EVENT. Input from devices is obtained in different ways depending on the mode as follows.

g)	REQUEST:	A specific invocation of REQUEST INPUT causes an attempt to read a logical input value from a specified logical input device. This can only occur when the logical input device is in REQUEST mode. GKS-N waits until the input is entered by the operator or a break action is performed by the operator. The break action is dependent on the logical input device and on the implementation. If a break occurs, the logical input value is not valid.
h)	SAMPLE:	A specific invocation of SAMPLE INPUT causes GKS-N, without waiting for an operator action, to return the current logical input value of a specified logical input device. This can only occur when the logical input device is in SAMPLE mode.
i)	EVENT:	GKS-N maintains one input queue containing temporally ordered

event reports. An event report contains the identification of a logical input device and a logical input value from that device. Event reports are generated asynchronously, by operator action only, from input devices in EVENT mode.

The application program can remove the oldest event report from the queue and examine its contents. The application can also flush from the queue all event reports from a specified logical input device.

A specific logical input device is said to be taking part in an interaction during the whole time that it is in SAMPLE or EVENT mode, but, when it is in REQUEST mode, only during the execution of a REQUEST INPUT function for that device. Alternatively, an interaction with the device may be said to be underway during that time. Many devices on many workstations may be taking part in interactions simultaneously.

6.7.2 Logical input device model

To describe the precise actions of the logical input devices, it is first necessary to describe their relationship with physical input devices, using the concept of measures and triggers.

A logical input device contains a measure, a trigger, an initial value, a prompt and echo type, an echo area and a data record containing details about the prompt and echo type. A logical input device's measure and trigger are parts of the implementation of the workstation containing the logical input device. Initial value, prompt and echo type, echo area, and data record can be supplied by the application program.

The measure of a logical input device is a value determined by one or more physical input devices together with a 'measure mapping'. More than one measure can simultaneously be determined by a single physical device; a separate measure mapping applies for each measure. A measure can be seen as the state of an independent, active process (a measure process). Each state corresponds exactly with a logical input value.

The current state of the measure process (i.e. the device's measure) is available to GKS-N as a logical input value. Whenever the device is taking part in an interaction, the measure process is in existence. Under other conditions, this process does not exist.

While the measure process is in existence, if echoing is required, output indicating the current state of the measure process is provided to the operator.

The trigger of a logical input device is a physical input device or a set of them together with a 'trigger mapping'. The operator can use a trigger to indicate significant moments in time. At these moments, the trigger is said to 'fire'. A single operator action (for example, pressing a button or a light pen tip switch) causes the firing of not more than one trigger. Several logical input devices can refer to the same trigger.

A trigger can be seen as an independent, active process (a trigger process) that sends a message to one or more recipients when it fires. A logical input device is a recipient of its trigger if there is a pending REQUEST for it or if it is in EVENT mode. Both of these conditions can be true simultaneously for different logical input devices. If there is at least one recipient for a trigger, the trigger process is in existence. Under other conditions this process does not exist.

If a REQUEST for a logical input device is pending when the device's trigger fires, the measure of that device is used to satisfy the REQUEST. If one or more devices containing a given trigger are in EVENT mode when the trigger fires, the identifications of those devices and their measure values are passed to the input queue mechanism as separate event reports. The input queue mechanism is described in detail in 6.7.5.

When a trigger firing succeeds in satisfying a REQUEST, or adding event records to the input queue, GKS provides to the operator an acknowledgement the form of which depends on the implementation of the logical input device. The acknowledgement is not controllable by a GKS function.

6.7.3 Operating modes of logical input devices

The mode of a logical input device may be changed by invoking the appropriate SET LOGICAL INPUT DEVICE MODE function.

After an invocation of SET LOGICAL INPUT DEVICE MODE with the parameter REQUEST, no measure process exists for the specified device and the device's identifier is not on its trigger's list of recipients. After an invocation with the parameter EVENT, a newly initiated measure process is in existence for the specified device and the device's identifier is on its trigger's list of recipients.

After an invocation with the parameter SAMPLE, a newly initiated measure process is in existence for the specified device, but the device's identifier is not on its trigger's list of recipients.

Initially a logical input device is in REQUEST mode. While a device is in REQUEST mode, a logical input value may be obtained by invoking the appropriate REQUEST INPUT function. The effects of doing so are as follows.

a) To create a measure process for the specified device and set its initial value. Echoing is performed by the measure process if echoing is on for the specified device.

b) To add the device's identifier to its trigger's list of recipients. If the list was previously empty, the trigger process is started.

c) To suspend GKS-N until the trigger of the specified device fires, or the operator invokes the break facility.

d) If the trigger fired, to set the logical input value to the current state of the measure process.

e) To destroy the measure process.

f) To remove the device's identifier from its trigger's list of recipients. If this list becomes empty, the trigger process is destroyed.

g) If the trigger fired, to return the logical input value and the status OK, otherwise to return the status NONE.

While a logical input device is in SAMPLE mode, a logical input value may be obtained by invoking the SAMPLE INPUT function. The effect of doing so is to set the logical input value to the current state of the measure process without waiting for a trigger firing.

While a logical input device is in EVENT mode, logical input values are added as event reports to the input queue, and may be obtained in sequence by invoking AWAIT INPUT.

Figure 9 shows the effect of every operating mode on the measure and trigger of a logical input device.

6.7.4 Measures of each logical input device class

Details of the measures of logical input devices of different classes are as follows.

A LOCATOR measure consists of a position in world coordinates (P) and a normalization transformation number (N). Then P transformed to NDC by N lies within the workstation window and within the viewport specified by N and outside all viewports of higher priority than N.

A STROKE measure consists of a sequence of points in world coordinates and a normalization transformation number (N). Let $P_1 \ldots P_m$ be the points. Then P_i $(1 \leq i \leq m)$ transformed to NDC by N lie within the workstation window and within the viewport specified by N and there is no viewport of higher priority than N containing all the points P_i transformed to NDC by N. Thus, N may change as points are added to the stroke.

Any invocation of SET WINDOW AND VIEWPORT or SET VIEWPORT INPUT PRIORITY can cause a change in P (any P_i for STROKE) or N or both, but the above conditions hold for the new values.

The rules imply that no normalization transformation having priority less than that of transformation 0 can appear in the state of a LOCATOR or STROKE measure process (with the default settings of the viewport input priorities, normalization transformation 0 has the highest).

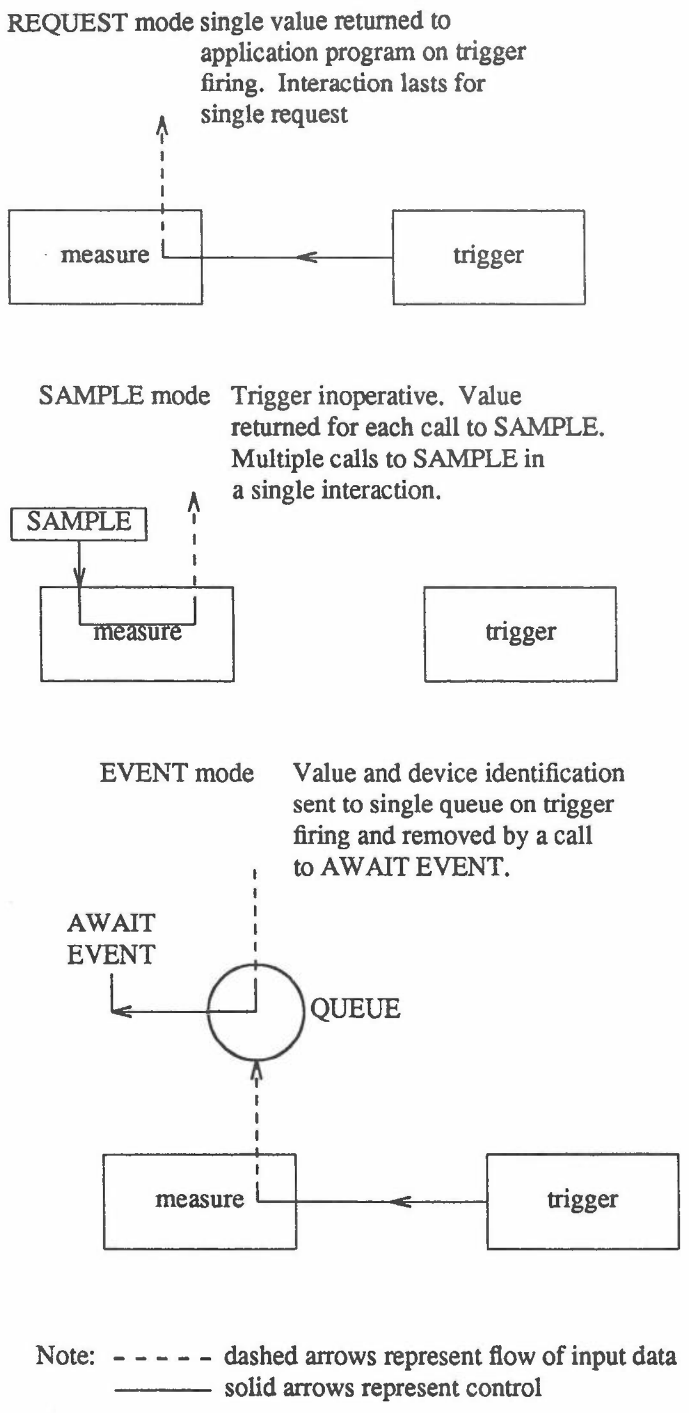

REQUEST mode single value returned to application program on trigger firing. Interaction lasts for single request
measure
trigger
SAMPLE mode Trigger inoperative. Value returned for each call to SAMPLE. Multiple calls to SAMPLE in a single interaction.
SAMPLE
measure
trigger
EVENT mode Value and device identification sent to single queue on trigger firing and removed by a call to AWAIT EVENT.
AWAIT EVENT
QUEUE
measure
trigger
Note: - - - - - dashed arrows represent flow of input data
——— solid arrows represent control

Figure 9 - Operating modes using measures and triggers

A VALUATOR measure provides logical input values that are real numbers. Each value lies between (possibly including) minimum and maximum values, which are in the data record in the workstation state list.

A CHOICE measure provides logical input values whose components are OK or NOCHOICE and an integer in the range 1 to a device dependent maximum specified in the workstation description table. If the first component is OK, then the integer is valid. CHOICE input typically occurs when an operator presses a button (the numeric identifier of the button determines the measure) or combination of buttons (the measure is derived from the combination of buttons pressed).

A PICK measure provides logical input values whose components are OK or NOPICK, NAMESET and a PICK IDENTIFIER. If the first component is OK, then the NAMESET and PICK IDENTIFIER obey the following rules:

a) The output primitive picked is visible and detectable on the workstation.

b) The NAMESET and PICK IDENTIFIER are the NAMESET and PICK IDENTIFIER attributes of the output primitive picked. Part of the primitive lies within the workstation window and, if clipping was on, part also lies within the primitive's set of clipping rectangles and, if shielding was on, part also lies outside the primitive's set of shielding rectangles.

The PICK initial value is tested against the above rules whenever the PICK measure process is initiated. If the rules are not satisfied, the process state is set to NOPICK.

A STRING measure provides logical input values which are character strings up to a device dependent maximum length specified by the buffer size value in the data record in the workstation state list.

6.7.5 Input queue and current event report

The input queue contains zero or more event reports. Event reports contain pairs of values (device identifier, logical input value) resulting from trigger firings. Event reports can be added to the input queue when logical input devices in EVENT mode are triggered by the operator. Events can be removed from the input queue by invocations of AWAIT INPUT, and FLUSH DEVICE EVENTS.

When a trigger that is part of one or more logical input devices in EVENT mode fires, the resulting event reports are entered into the queue. A set of event reports, one for each device is added to the input queue, if and only if there is room for the whole set of simultaneous event reports. AWAIT INPUT returns all the simultaneous event reports together.

If there is not room in the queue for all event reports when a trigger fires, input queue overflow has occurred. Input queue overflow is not reported to the application program immediately. It is reported via the error mechanism during the next invocation of any GKS-N function that can remove event reports from the input queue (AWAIT INPUT, FLUSH DEVICE EVENTS, and CLOSE WORKSTATION). The input queue has to be emptied before further event reports will be added. Between the detection of input queue overflow and the next time AWAIT INPUT is invoked with the input queue empty, no events are generated by trigger firings and thus no acknowledgements are provided. (This permits the application program to determine how many events were in the queue when overflow occurred by calling AWAIT INPUT with zero timeout.)

When the 'input queue overflow' error is reported, the trigger causing the overflow is indicated by placing into the error state list the identification of any one of the logical input devices using that trigger which was in EVENT mode at the time the overflow was detected.

AWAIT INPUT, if the queue is not empty, removes the first event report and returns it to the application program. If the queue is empty, AWAIT INPUT suspends execution until an event report is queued or until the specified timeout period has elapsed.

FLUSH DEVICE EVENTS removes all event reports for a specific device from the input queue.

6.7.6 Transformation of LOCATOR and STROKE input

The application programmer requires LOCATOR input to define a position in the most appropriate world coordinate system currently defined by the set of normalization transformations. This is achieved by first transforming the input data from DC to NDC by the inverse workstation transformation which is in effect when LOCATOR input is generated (see 8.10). LOCATOR input always defines a position in the NDC range $[0,1]\times[0,1]$.

To return to the application program a position in world coordinates, the position in NDC space needs to be transformed from NDC to WC by the inverse of one of the normalization transformations. Each normalization transformation has associated with it a viewport input priority which is only relevant to LOCATOR and STROKE input. Normalization transformations are ordered in a list defined by the viewport input priority. At GKS initialization, an implementation defined number of normalization transformations are initialized to have window and viewport set to the unit square and their viewport input priorities are set relative to the transformation number with transformation number 0 given the highest priority, transformation number 1 the next highest and so on. Changing the viewport input priority of any normalization transformation is allowed at any time.

The LOCATOR input position in NDC space is compared with the viewports of the normalization transformations, to find the normalization transformation with the viewport which has the highest viewport input priority and contains the LOCATOR position. The LOCATOR position is transformed by the inverse of this normalization transformation to the associated WC. This LOCATOR position is returned to the application program in WC together with the number of the normalization transformation used.

As transformation number 0 is the unity transformation with viewport $[0,1]\times[0,1]$ and cannot be changed, this ensures that LOCATOR input is always within at least one viewport. A data flow chart for LOCATOR input is given in figure 10.

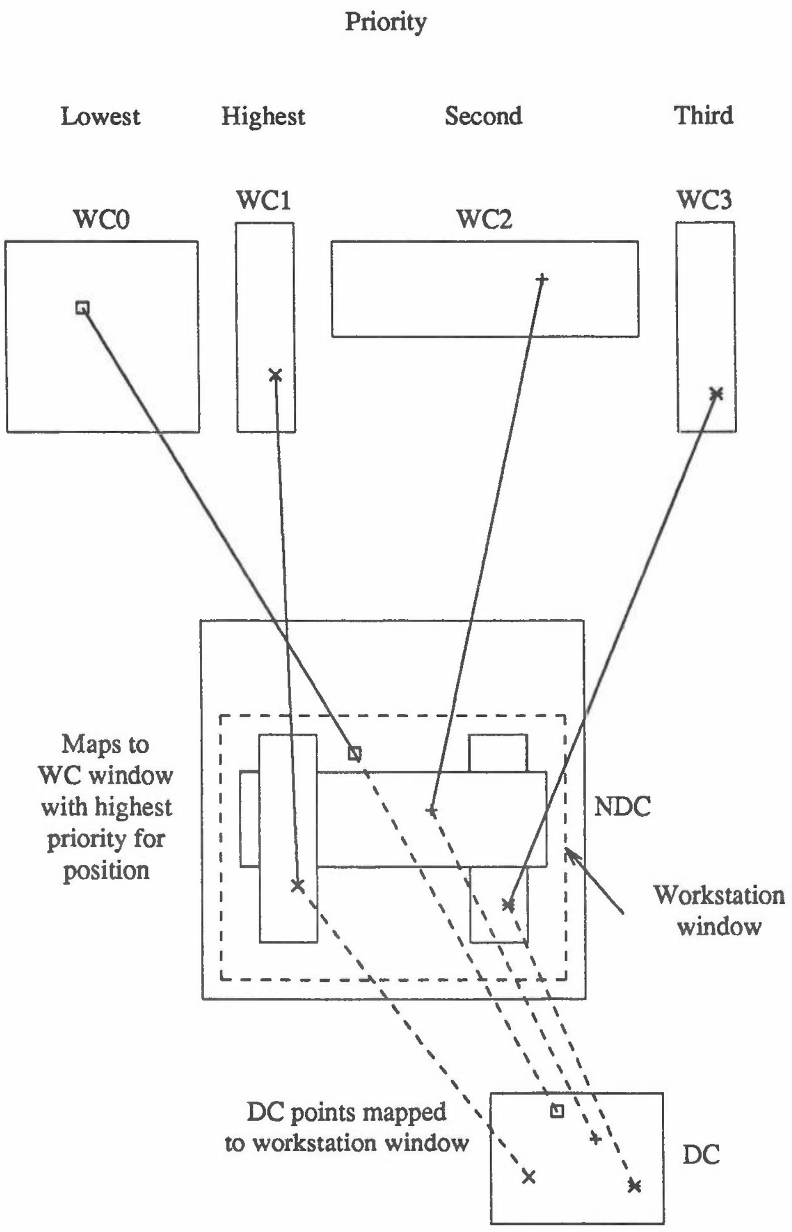

Priority
Lowest
Highest
Second
Third
WC0
WC1
WC2
WC3
Maps to WC window with highest priority for position
NDC
Workstation window
DC points mapped to workstation window
DC

Figure 10 - LOCATOR input

As transformation number 0 is given the highest viewport input priority initially, LOCATOR input is effectively returned in WC equivalent to NDC until a normalization transformation is defined with a viewport input priority greater than that of transformation number 0. If a normalization transformation is no longer required for mapping LOCATOR input back to WC, it can effectively be hidden by reassigning it a viewport input priority lower than transformation number 0.

Changing the viewport input priority of transformation number 0 is allowed.

In an event report, generated by a LOCATOR device in EVENT mode, the DC position is transformed to the appropriate WC position before the event report is placed on the input queue. These transformations may be performed while the normalization and workstation transformations are being changed; thus, there is a race condition. The implementation has therefore to treat the transformations as resources to be allocated and deallocated between the competing processes.

Similar considerations apply to transformation of STROKE input as apply to LOCATOR input, with the complication that more than one point is involved.

When each point of a stroke is generated, the coordinates of the point are transformed from DC to NDC by the inverse workstation transformation then in effect (see 8.11). STROKE input always consists of points in the NDC range [0,1]×[0,1].

The STROKE points in NDC space are compared with the viewports of the normalization transformations, to find the normalization transformation with the viewport which has the highest viewport input priority and contains all of the points. The STROKE points are then transformed by the inverse of this normalization transformation and returned to the application program in WC together with the number of the normalization transformation used.

If the STROKE device is in SAMPLE mode, the normalization transformation used may vary between successive samples.

In EVENT mode, there is a similar race condition to that applying to LOCATOR input. Between placing an event report on the input queue and the execution of AWAIT EVENT which removes the STROKE event from the queue, it is possible for the normalization transformation and the workstation transformation to be changed by the application program. To ensure that the DC points input by the operator are equivalent to the WC points retrieved from the input queue, it is advisable for the application program not to change transformations while a STROKE device is in EVENT mode.

6.8 Clipping and shielding

Associated with output primitives in the NDC picture are a set of clipping rectangles and a set of shielding rectangles. The current set of each are added to the output primitive as it is added to the NDC picture.

Invoking the functions SET CLIPPING RECTANGLE SET and SET SHIELDING RECTANGLE SET define a set of clipping and shielding rectangles. These sets can be made active by invoking SET CLIPPING INDICATOR with the parameter CLIP or SET SHIELDING INDICATOR with the parameter SHIELD.

Clipping and shielding occurs as the NDC picture is transformed and rendered by the workstation.

6.9 GKS-N levels

The GKS-N system is designed for use in an interactive environment. GKS-N provides two levels. The first level just contains the geometric primitives while the second level contains both the geometric primitives and bundled primitives together with all the relevant attributes and workstation control.

6.10 Inquiry functions

Inquiry functions return values directly from or derived from the various state lists and description tables. The data types of the values and the default values of the entries are summarized in clause 10.

The inquiry functions of GKS-N are designed in such a way that they do not cause any errors to be generated. Inquiry functions for values that may be logically unavailable have an output parameter, 'error indicator', that determines whether or not the other returned values are valid. The availability parameter is of type integer and, in the event of the other values not being available, returns an error number, which identifies the appropriate GKS-N error condition. The same error numbers are used as for non-inquiry functions and thus the standard list of error messages should be consulted. If GKS-N is not in the proper state, then the error number appropriate to this condition is the one returned, even if there are other reasons for the values being unavailable. If the values are available, zero is returned in the error indicator parameter.

For all values except zero the returned output values are implementation dependent. The description of each inquiry function lists the error indicator values that the function can return.

6.11 Error handling

For each GKS-N function, a finite number of error situations is specified, any of which will cause the ERROR HANDLING function to be invoked. The ERROR HANDLING function defined in GKS-N invokes an ERROR LOGGING function which appends an error message and identification of the GKS-N function (which caused the error) to the error file before returning to the ERROR HANDLING function.

This two-stage invocation of error handling allows the ERROR HANDLING function provided to be replaced by the application while still having access to services provided by the ERROR LOGGING function.

All GKS-N functions check that it is used appropriately and that the values of input parameters are valid before attempting to execute the function. At least the first error detected will cause the ERROR HANDLING function to be invoked.

The application supplied ERROR HANDLING function is only allowed to invoke GKS-N inquiry functions, the ERROR LOGGING function and the EMERGENCY CLOSE GKS function. Inquiry functions are not allowed to generate errors.

If the application detects errors outside GKS-N and regains control, it can invoke the EMERGENCY CLOSE GKS function which will attempt to save as much of the graphical information as possible. GKS-N itself may invoke the EMERGENCY CLOSE GKS function if it gets into difficulty.

6.12 Special interfaces between GKS and the application program

A uniform escape mechanism for allowing access to installation and hardware specific features (a 'standard way of being non-standard') is provided by means of the ESCAPE function. Although the use of this mechanism reduces the portability of the application program, it does so in an easily identifiable manner.

The ESCAPE function does not generate geometrical output; by contrast, the GENERALIZED DRAWING PRIMITIVE can generate geometrical output not otherwise generated by GKS.

7 GKS functions

7.1 Notational conventions

The heading of each function specifies

a) the function's name;

b) references to the function (aligned to the right).

The GKS functional capabilities are summarized in annex 8.5.

The parameter lists indicate for each entry

c) whether the entry is an input (In) or output (Out) parameter (aligned to the left);

d) the name of the parameter;

e) the data type (aligned to the right).

The first three type definitions define sequences of an arbitrary type (X) of length greater than 1, length greater than 2 and length greater than 3, respectively. The ordinary sequence type seq X, allows sequences of length 0.

$\mathrm{seq}_1\ X == \{f : \mathrm{seq}\ X \mid \#f > 1\}$
$\mathrm{seq}_2\ X == \{f : \mathrm{seq}\ X \mid \#f > 2\}$
$\mathrm{seq}_3\ X == \{f : \mathrm{seq}\ X \mid \#f > 3\}$

Points in world coordinates are of type WCPoint.

$\mathrm{WCPoint} == \mathbf{R} \times \mathbf{R}$

Character strings are sequences of characters. Character is a basic type.

Char
CharString == seq Char

Colour indices are of type ColrInd. This type is a basic type.

ColrInd

Colour index arrays are of type ColArray.

$\mathrm{ColArray} == \mathbf{N} \times \mathbf{N} \rightarrow \mathrm{ColrInd}$

Types for bundled primitives are now defined.

Prim	::= POLYLINE | POLYMARKER | TEXT | FILLAREA | CELLARRAY | GDP
VBPrimParam	::= polyline <<seq_2 WCPoint>> | polymarker <<seq_1 WCPoint>> | text <<WCPoint × CharString>> | fillarea <<seq_3 WCPoint>> | cellarray <<WCPoint × WCPoint × ColArray>> | gdp <<seq WCPoint × GDPId × GDPData>>

Types forGDP identifiers and data records are base types.

GDPId
GDPData

The type EFile is used in OPEN GKS for the type of the error file.

EFile

The values of the type PrimAttr are the names of the primitive attributes. These are listed in the table below together with the types of the values of each attribute.

Attribute name	Type
POLYLINE INDEX	PolylineInd
LINETYPE	LineType
LINEWIDTH SCALE FACTOR	LineWidth
POLYLINE COLOUR INDEX	ColrInd
POLYMARKER INDEX	PolymarkerInd
MARKERTYPE	MarkerType
MARKER SIZE SCALE FACTOR	MarkerSize
POLYMARKER COLOUR INDEX	ColrInd
TEXT INDEX	TextInd
TEXT FONT AND PRECISION	FontPrec
CHARACTER EXPANSION FACTOR	Expan
CHARACTER SPACING	Spacing
TEXT COLOUR INDEX	ColrInd
CHARACTER HEIGHT	CharHeight
CHARACTER UP VECTOR	CharUpVector
TEXT PATH	Path
TEXT ALIGNMENT	Align
FILL AREA INDEX	FillInd
FILL AREA INTERIOR STYLE	FillInterior
FILL AREA STYLE INDEX	FillStyleInd
FILL AREA COLOUR INDEX	ColrInd
PATTERN SIZE	PatternSize
PATTERN REFERENCE POINT	WCPoint
PICK IDENTIFIER	PickId
NAMESET	NameSet

Of the types listed in the table above, the following are basic types.

PolylineInd	
LineType	$== \{x{:}N \mid x \neq 0\}$
ColrInd	
PolymarkerInd	
MarkerType	$== \{x{:}N \mid x \neq 0\}$
TextInd	
FillInd	
FillStyleInd	
PickId	

Definitions of the remaining types are given below.

LineWidth	$== \{x:\mathbf{R} \mid x \geq 0\}$
MarkerSize	$== \{x:\mathbf{R} \mid x \geq 0\}$
FontPrec	$==$ Font $\times$ Prec
Font	$== \{x:\mathbf{N} \mid x \neq 0\}$
Prec	::= STRINGPREC | CHARPREC | STROKEPREC
Expan	$== \{x:\mathbf{R} \mid x \geq 0\}$
Spacing	$== \mathbf{R}$
CharHeight	$== \{x:\mathbf{R} \mid x \geq 0\}$
CharUpVector	$== \mathbf{R} \times \mathbf{R}$
Path	::= RIGHT | LEFT | UP | DOWN
Align	$== \{x : \text{Horizontal}, y : \text{Vertical} \bullet (x,y)\}$
Horizontal	::= HNORMAL | LEFT | CENTRE | RIGHT
Vertical	::= VNORMAL | TOP | CAP | HALF | BASE | BOTTOM
FillInterior	::= HOLLOW | SOLID | PATTERN | HATCH
PatternSize	$== \{x,y:\mathbf{R} \mid x>0 \wedge y>0 \bullet (x,y)\}$
NameSet	$==$ **P** Name
	Name is a basic type

The type VPrimAttr is the supertype of all the types of the primitive attributes.

```
VPrimAttr ::= polylineind <<PolylineInd>> |
              linetype <<LineType>> |
              linewidth <<LineWidth>> |
              colrind <<ColrInd>> |
              polymarkerind <<PolymarkerInd>> |
              markertype <<MarkerType>> |
              markersize <<MarkerSize>> |
              textind <<TextInd>> |
              fontprec <<FontPrec>> |
              expan <<Expan>> |
              spacing <<Spacing>> |
              charheight <<CharHeight>> |
              charupvector <<CharUpVector>> |
              path <<Path>> |
              align <<Align>> |
              fillind <<FillInd>> |
              fillinterior <<FillInterior>> |
              fillstyleind <<FillStyleInd>> |
              patternsize <<PatternSize>> |
              refpoint <<WCPoint>> |
              asfs <<Asfs>> |
              pickid <<PickId>> |
              nameset <<NameSet>>
```

The type LogAttr is a subtype of PrimAttr and includes just the names of the logical attributes of bundled primitives.

LogAttr	::= LINETYPE | LINEWIDTH SCALE FACTOR |POLYLINE COLOUR INDEX | MARKERTYPE | MARKER SIZE SCALE FACTOR | POLYMARKER COLOUR INDEX | TEXT FONT AND PRECISION | CHARACTER EXPANSION FACTOR | CHARACTER SPACING | TEXT COLOUR INDEX | FILL AREA INTERIOR STYLE | FILL AREA STYLE INDEX | FILL AREA COLOUR INDEX

The type VAsf represents values of attribute source flags.

VAsf	::= BUNDLED | INDIVIDUAL

The following types are used to describe transformations and clipping control.

NormTran	
NormTran1	
WCWindow	$== \{x_{\min}, x_{\max}, y_{\min}, y_{\max}: \mathbf{R} \mid x_{\min}<x_{\max} \wedge y_{\min}<y_{\max} \bullet (x_{\min}, x_{\max}, y_{\min}, y_{\max})\}$
NDCViewport	$== \{x_{\min}, x_{\max}, y_{\min}, y_{\max}: \mathbf{R} \mid x_{\min}<x_{\max} \wedge y_{\min}<y_{\max}$ $\wedge\ 0 \le x_{\min} < 1$ $\wedge\ 0 < x_{\max} \le 1$ $\wedge\ 0 \le y_{\min} < 1$ $\wedge\ 0 < y_{\max} \le 1 \bullet (x_{\min}, x_{\max}, y_{\min}, y_{\max})\}$
Rectangle	$== \{x_{\min}, x_{\max}, y_{\min}, y_{\max}: \mathbf{R} \mid x_{\min}<x_{\max} \wedge y_{\min}<y_{\max}$ $\wedge\ 0 \le x_{\min} < 1$ $\wedge\ 0 < x_{\max} \le 1$ $\wedge\ 0 \le y_{\min} < 1$ $\wedge\ 0 < y_{\max} \le 1 \bullet (x_{\min}, x_{\max}, y_{\min}, y_{\max})\}$
RectangleSet	== P Rectangle
Priority	::= HIGHER | LOWER
Vip	::= NormTran $\rightarrow$ **N**
Clip	::= CLIP | NOCLIP
Shield	::= SHIELD | NOSHIELD

NDCWindow	$== \{x_{min}, x_{max}, y_{min}, y_{\max}: \mathbf{R} \mid x_{\min}<x_{\max} \wedge y_{\min}<y_{\max} \bullet (x_{\min}, x_{\max}, y_{\min}, y_{\max})\}$
DCViewport	$== \{x_{min}, x_{max}, y_{min}, y_{\max}: \mathbf{R} \mid x_{\min}<x_{\max} \wedge y_{\min}<y_{\max} \bullet (x_{\min}, x_{\max}, y_{\min}, y_{\max})\}$

The type NormTran1 is a subtype of NormTran and excludes normalization transformation number 0.

The type PartName is a basic type for the names of picture parts.

PartName

The type ImageFunction is a basic type describing images.

ImageFunction

Values of the names associated with picture parts in archives are of type ArchName. This is a basic type.

ArchName

NDC archive identifiers are of type ArchId.

ArchId

The type Matrix23 denotes a 2×3 matrix.

Matrix23 $== \{i,j{:}\mathrm{N},\ m{:}\mathrm{R} \mid 1 \le i \le 2,\ 1 \le j \le 3 \bullet (i,j) \rightarrow m\}$

Types for input are described below.

MeasureId	
TriggerId	
MeasureClass	::= LOCATOR | STROKE | VALUATOR | CHOICE | PICK | STRING | COMPOSITE
DeviceId	== WsId × $\mathbf{N}_1$ × MeasureClass
Mode	::= REQUEST | SAMPLE | EVENT
InputStatus	::= OK | NONE
ChoiceStatus	::= OK | NOCHOICE
PickStatus	::= OK | NOPICK
InputValue	::= locatorvalue <<NormTran × WCPoint>>
	| strokevalue <<NormTran × seq WCPoint>>
	| valuatorvalue <<**R**>>
	| choicevalue <<ChoiceStatus × N>>
	| pickvalue <<PickStatus × NameSet × PickId>>
	| stringvalue <<CharString>>
	| compvalue <<CompValue>>
InitialInput	== InputValue × PEType × Echo × EchoArea × InputData
Echo	::= ECHO | NOECHO
EchoArea	== $\{x_{min}, x_{max}, y_{min}, y_{max}: \mathbf{R} \mid x_{min}<x_{max} \wedge y_{min}<y_{max} \bullet (x_{min}, x_{max}, y_{min}, y_{max})\}$
Timeout	== $\{x{:}\mathbf{R} \mid x \geq 0\}$

The type PEType is a basic type describing prompt and echo types and InputData is a basic type describing input data records.

Values of the identifiers associated with pictures in metafiles are of type PictureId. This is a basic type.

PictureId

NDC metafile identifiers are type NDCMetafileID and rendered metafile identifiers are of type RenderFileId

NDCMetafileId
RenderFileId

Types associated with workstation control are listed in the table below. All are basic types.

Entity	Type
Workstation identifier	WsId
Workstation type	WsType

The types associated with bundles are listed below.

Bundle	Type
Polyline	LineBundle
Polymarker	MarkerBundle
Text	TextBundle
Fill area	FillBundle

Type definitions are given below.

LineBundle	== LineType × LineWidth × ColrInd
MarkerBundle	== MarkerType × MarkerSize × ColrInd
TextBundle	== FontPrec × Expan × Spacing × ColrInd
FillBundle	== FillInterior × FillStyleInd × ColrInd

In addition, pattern arrays are defined as follows.

PatternArray	== $\{i,j{:}\mathbf{N},\ m{:}\ \text{ColrInd} \mid 1 \leq i \leq \max_x,\ 1 \leq j \leq \max_y \bullet (i,j) \rightarrow m\}$

Colour models are specified by values of the type ColMod:

ColMod	== $\{x:\mathrm{N} \mid x \neq 0\}$

The type Colr specifies colour coordinates. The number of coordinates necessary is dependent on the colour model. Colours in the RGB and CIELUV colour models, which are required to be available, are specified by three coordinates and the type definitions are:

ColRGB	== $\{r,g,b:\mathrm{R} \mid 0 \leq r \leq 1 \wedge 0 \leq g \leq 1 \wedge 0 \leq b \leq 1 \bullet (r,g,b)\}$
ColCIELUV	== $\{u',v',Y:\mathrm{R} \bullet (u',v',Y)\}$
Colr	::= colrgb <<ColRGB>> | colcieluv <ColCIELUV>>

The type LCC specifies luminance values and chromaticity coefficients.

LCC	== $\{u',v',Y:\mathrm{R} \bullet (u',v',Y)\}$

The bundle, pattern and colour tables are described by the types:

Representation	::= POLYLINE | POLYMARKER | TEXT | FILLAREA | PATTERN | COLOUR
VRep	::= linebundle <<LineBundle>> | markerbundle <<MarkerBundle>> | textbundle <<TextBundle>> | fillbundle <<FillBundle>> | patternarray <<PatternArray>> | colr <<Colr>>
Rep	::= polylineind <<PolylineInd>> polymarkerind <<PolymarkerInd>> | textind <<TextInd>> | fillind <<FillInd>> | fillstyleind <<FillStyleInd>> | colrind <<ColrInd>>
PolylineBT	== PolylineInd → LineBundle
PolymarkerBT	== PolymarkerInd → MarkerBundle
TextBT	== TextInd → TextBundle
FillBT	== FillInd → FillBundle
PatternBT	== FillStyleInd → PatternArray
ColourT	== ColrInd → Colr

Workstation selection criteria are defined by the types:

SelectCrit	
SelectType	::= VISIBILITY | HIGHLIGHTING | DETECTABILITY
SelectT	== SelectType → SelectCrit

The next type is used by inquiry functions.

RType	::= SET | REALIZED

The following types describe the operating state of GKS. The first distinguishes GKS closed from GKS open. The second distinguishes the state in which there is no open picture part from the state in which a picture part is open. The third distinguishes the state in which there is no shape open from that in which there is a shape open; the fourth distinguishes closed path closed from closed path open.

GKSOpSt	::= GKCL | GKOP
PPOpSt	::= PPCL | PPOP
SHOpSt	::= SHCL | SHOP
CPOpSt	::= CPCL | CPOP

The following types are used by the ESCAPE function

EscapeId
InEscapeData
OutEscapeData

The following types are used by the error handling functions:

ErrNum
FName (elabourate to all GKS-N function names later)

The following types are used to describe shapes and paths.

ShapeName	
InsideRule	::= EVENODD | WINDING
SubPathClass	::= LINES | CURVES | ARC | ELLIPTICALARC | NUB
SplineOrder	== $\{i{:}\mathrm{N} \mid i>0\}$
ControlPoints	== seq_1 WCPoint
Knots	== seq_1 **R**
ParamRan	== **R**×**R**
NUB	== { s: SplineOrder, cp: ControlPoints, kn: Knots, $(t_{\min},t_{\max})$:ParamRan | $\#kn = s+\#cp \land (\forall\ j,k \in 1..\#kn \mid k>j,\ kn(k) \geq kn(j)) \land$ $t_{\min} > kn(s) \land t_{\max} < kn(\#cp+1) \land t_{\min} < t_{\max}$ $\bullet\ (s,cp,kn,(t_{\min},t_{\max}))\}$
SubPathParam	::= lines <<seq_2 WCPOINT>> | curves <<WCPOINT × WCPOINT × WCPOINT × WCPOINT>> | arc <<WCPOINT × WCPOINT × WCPOINT × **R**>> | ellipticalarc <<WCPOINT × WCPOINT × WCPOINT>> | nub <<NUB>>
PathAttr	::= STYLE | WIDTH | CAP | JOIN
Style	::= SOLID | DOTTED | DASHED
CoordSyts	::= WC | NDC
Coord	::= X | Y
Width	::= CoordSys × Coord × **R**
Cap	::= BUTTED | ROUNDED | SQUARE
Join1	::= ROUND | BEVEL
Join2	::= MITRED
Join	::= join1 <<Join1>> | join2 <Join2 × R>>
VPathAttr	::= style <<Style>> | width <<Width>> | cap <<Cap>> | join <<Join>>
ShapeAttr	::= TOPY | CAPY | HALFY | BASEY | BOTTOMY | CENTREY | LEFTX | RIGHTX | CENTREX | CENTRE | ORIGIN | TOP | BOTTOM | LEFT | RIGHT | TOPLEFT | TOPRIGHT | BOTTOMLEFT | BOTTOMRIGHT | START | MIDDLE | END
ConcatCrit	

7.2 Control functions

OPEN GKS 6.11

In	error file	File

The GKS operating state is set to GKOP = 'GKS open'. The GKS state list is allocated and initialised as indicated in 10. The GKS description table and the workstation description tables are made available. The entry 'error file' in the GKS error state list is set to the value specified by the parameter.

CLOSE GKS 6.11

none

The GKS operating state is set to GKCL = 'GKS closed'. The GKS description table, GKS state list and the workstation description tables become unavailable. GKS can be reopened by invoking the function OPEN GKS.

ESCAPE 6.12

In	specific escape function identification	EscapeId
In	escape input data record	InEscapeData
Out	escape output data record	OutEscapeData

The specified non-standard specific escape function is invoked. The form of the escape data records and which of them are used may vary for different functions. The following rules govern the initial definition of a specific escape function:

a) the GKS concepts (see clause 5) are not violated;

b) the GKS state lists are not altered;

c) the function does not generate graphical output;

d) any side effects are well documented.

Specific escape functions may apply to more than one workstation, for example all open workstations. The escape input data record can include a workstation identifier where this is required. Examples of specific escape functions are:

e) local control of a frame buffer;

f) use of specialist hardware.

g) switch between alphanumeric and graphics modes.

Specific escape function identifications are registered in the ISO International Register of Graphical Items, which is maintained by the Registration Authority (see 4.2). When a specific escape function has been approved by ISO, the specific escape function identification will be assigned by the Registration Authority.

EMERGENCY CLOSE GKS 6.11

none

GKS is closed due to an emergency. All open workstations are closed and GKS is closed. This function may be called even if there has already been an error. If GKS is already closed, no action is taken.

ERROR HANDLING		6.11
In	error number as listed in annex B	ErrNum
In	identification of the GKS function which caused the error detection	FName
In	error file	File

The ERROR HANDLING function is invoked by GKS in any of the error situations listed in Annex B. The standard function just calls the ERROR LOGGING function with the same parameters. The ERROR HANDLING function may be replaced by an application program supplied function to allow specific reaction to some error situations.

ERROR LOGGING		6.11
In	error number as listed in annex B	ErrNum
In	identification of the GKS function which caused the error detection	FName
In	error file	File

The ERROR LOGGING function appends an error message and identification of the GKS function that caused the error to the error file and then returns to the calling function.

7.3 Output functions

CREATE BUNDLED PRIMITIVE		6.2.9, 8.6, 8.7
In	primitive type	Prim
In	primitive parameters	VBPrimParam

A primitive of the specified type is created. The current values of the primitive's attributes as given by the GKS state list (see 10.2.3) are bound to the primitive.

If there is no picture part open, the current sets of clipping and shielding rectangles are also bound to the primitive and the primitive is added to the NDC picture.

If there is a picture part open, the primitive is stored in the open picture part. No sets of clipping or shielding rectangles are bound to the primitive and the primitive is not added to the NDC picture.

CREATE GEOMETRIC PRIMITIVE		6.2.2
In	shape name	ShapeName
In	shape origin	WCPoint
In	shape transformation	Matrix23
In	picture part name	PartName
In	picture part transformation	Matrix23

A primitive of the specified class is created. The current values of the NAMESET and PICK IDENTIFIER attributes are bound to the primitive.

If there is no picture part open, the current sets of clipping and shielding rectangles are also bound to the primitive and the primitive is added to the NDC picture.

If there is a picture part open, the primitive is stored in the open picture part. No sets of clipping and shielding rectangles are bound to the primitive and the primitive is not added to the NDC picture.

BEGIN SHAPE 6.2.3, 6.2.4, 6.2.5, 6.2.6, 6.2.7, 6.2.8

In	shape name	ShapeName
In	winding rule	InsideRule

GKS is set into the operating state SHOP = 'Shape open'. The shape name is recorded as the 'name of the open shape' in the GKS state list (see 10.2.3). All subsequent closed path definitions until the next END SHAPE will become part of the shape definition. Any shape attributes set after the invocation of BEGIN SHAPE and before the invocation of END SHAPE will become attributes of the shape being defined.

END SHAPE 6.2.3

none

GKS is set into the operating state SHCL = 'Shape closed'. Closed paths may no longer be added to the previous, open shape. The 'name of the open shape' in the GKS state list (see 10.2.3) becomes unavailable for inquiry.

DELETE SHAPE 6.2.3

In	shape name	ShapeName

The specified shape is deleted. The shape name may be reused by the application.

BEGIN CLOSED PATH 6.2.4

none

GKS is set into the operating state CPOP = 'Closed path open'. All subsequent sub-path definitions until the next END CLOSED PATH will be collected in sequence into this closed path.

END CLOSED PATH 6.2.4

none

GKS is set into the operating state CPCL = 'Closed path closed'. Sub-paths may no longer be added to the previously-open closed path.

CREATE SUB PATH 6.2.4

In	sub-path class	SubPathClass
In	sub-path parameters	SubPathParam

A sub-path of the specified class is created and added to the sequence of sub-paths defining a closed path.

CREATE CONTOUR 6.2.6

In	shape name	ShapeName
In	sub-path class	SubPathClass
In	sub-path parameters	SubPathParam

A shape is created which is the contour of the sub-path specified by the path class and path parameters. The precise form of the contour is defined by the current attributes of the sub-path.

SET SUB PATH ATTRIBUTE 6.2.6

In	attribute name	SubPathAttr
In	attribute value	VSubPathAttr

The current entry in the GKS state list corresponding to the specified attribute name is set to the value specified by the attribute value. Attribute names and their types are listed in 7.1.

SET SHAPE ATTRIBUTE 6.2.7

In	attribute name	ShapeAttr
In	attribute value	VShapeAttr

The current attribute entry in the GKS state list corresponding to the specified attribute name is set to the value specified by the attribute value. The predefined attribute names and their types are listed in 7.1.

CREATE CONCATENATED SHAPE 6.2.8

In	shape name	ShapeName
In	sequence of shape names	seq_2 ShapeName
In	concatenation criterion	ConcatCrit

A new shape is defined with name equal to the first parameter. The new shape is a concatenation of the sequence of shapes defined by the sequence of shape names using the concatenation criterion to specify the geometric relationship between pairs of shapes in the shape name sequence.

7.4 Output attributes

SET PRIMITIVE ATTRIBUTE 6.2.1, 8.8

In	attribute name	PrimAttr
In	attribute value	VPrimAttr

The current attribute entry in the GKS state list corresponding to the specified attribute name is set to the value specified by the parameter. Attribute names and their types are listed in 7.1.

SET ATTRIBUTE SOURCE FLAG 6.2.1, 8.6

In	attribute name	PrimAttr
In	atrribute source flag value	VAsf

The current attribute source flag entry in the GKS state list corresponding to the specified attribute name is set to the value specified by the parameter.

7.5 Normalization transformation functions

SET WINDOW AND VIEWPORT 6.3, 6.7.4

In	transformation number	NormTran1
In	window limits	WCWindow
In	viewport limits	NDCViewport

The window and viewport limits entries of the specified normalization transformation in the GKS state list are set to the values specified by the parameters.

SET CLIPPING RECTANGLE SET 6.8

In	set of clipping rectangle limits	RectangleSet

The 'clipping rectangle set' entry in the GKS state list is set to the value specified by the parameter.

SET SHIELDING RECTANGLE SET 6.8

In	set of shielding rectangle limits	RectangleSet

The 'shielding rectangle set' entry in the GKS state list is set to the value specified by the parameter.

SET VIEWPORT INPUT PRIORITY 6.7.4, 8.10, 8.11

In	transformation number	NormTran
In	reference transformation number	NormTran
In	relative priority	Priority

The viewport input priority of the specified normalization transformation in the GKS state list is set to the next higher or next lower priority relative to the reference transformation according to the specified relative priority. If the specified transformation number is the same as the reference transformation number, the function has no effect.

SELECT NORMALIZATION TRANSFORMATION 6.3

In	transformation number	NormTran

The 'current normalization transformation number' entry in the GKS state list is set to the value specified by the parameter.

SET CLIPPING INDICATOR		6.7.4, 6.8
In	clipping indicator	Clip

The 'clipping indicator' entry in the GKS state list is set to the value specified by the parameter.

SET SHIELDING INDICATOR		6.7.4, 6.8
In	shielding indicator	Shield

The 'shielding indicator' entry in the GKS state list is set to the value specified by the parameter.

7.6 NDC picture functions

DELETE PRIMITIVES		6.5.2
In	selection criterion	SelectCrit

Primitives in the NDC picture whose nameset attributes satisfy the specified selection criterion are deleted from the NDC picture.

REMOVE NAME FROM NDC PICTURE		6.4.2
In	name	Name
In	selection criterion	SelectCrit

The specified name is removed from the namesets of all primitives in the NDC picture which satisfy the selection criterion.

ADD NAME TO NDC PICTURE		6.5.2
In	name	Name
In	selection criterion	SelectCrit

The specified name is added to the namesets of all primitives in the NDC picture which satisfy the specified selection criterion.

7.7 Metafile functions

COPY NDC PICTURE TO NDC METAFILE		6.5.3
In	metafile identifier	NDCMetafileId
In	picture identifier	PictureId
In	selection criterion	SelectCrit
In	nameset	NameSet

Primitives in the NDC picture which satisfy the specified selection criterion are stored in the specified metafile. The picture is given the specified picture identifier in the metafile. The names in the specified nameset are added to the nameset attribute of each primitive in the specified metafile.

COPY NDC METAFILE PICTURE TO NDC PICTURE 6.5.3

In	metafile identifier	NDCMetafileId
In	picture identifier	PictureId
In	nameset	NameSet

The specified picture in the specified metafile is added to the NDC picture. The names in the specified nameset are added to the nameset attribute of each primitive in the specified picture.

7.8 Picture part store functions

BEGIN PICTURE PART 6.4.1

In	picture part name	PartName

The picture part operating state is set to PPOP = 'Picture part open'. The picture part name is recorded as the 'name of the open picture part' in the GKS state list (see 7.6). All subsequent output primitives until the next END PICTURE PART will be collected in sequence into this picture part after the original contents. The current attributes are bound to the primitives. Sets of clipping and sets of shielding rectangles are not bound and the primitives are not added to the NDC picture.

END PICTURE PART 6.4.1

none

The picture part operating state is set to PPCL = 'Picture part closed'. Primitives may no longer be added to the previously open picture part. The 'name of the open picture part' in the GKS state list (see 10) becomes unavailable for inquiry.

ARCHIVE PICTURE PART 5.10

In	picture part name	PartName
In	archive identifier	ArchId
In	archive name	ArchName

The specified picture part is stored in the specified archive. The archive name is given to the picture part.

RETRIEVE PICTURE PART FROM ARCHIVE 5.10

In	archive identifier	ArchId
In	archive name	ArchName
In	picture part name	PartName

The picture part with archive name is retrieved from the specified archive and added to picture part store with the specified picture part name.

BEGIN PICTURE PART AGAIN 6.4.2

In	picture part name	PartName

The specified picture part is reopened. The picture part operating state is set to PPOP = 'Picture part open'. The picture part name is recorded as the 'name of the open picture part' in the GKS state list (see 10.2.3). All subsequent output primitives until the next END PICTURE PART will be collected in sequence into this picture part after the initial contents. The current attributes are bound to the primitives. Sets of clipping and sets of shielding rectangles are not bound and the primitives are not added to the NDC picture.

APPEND PICTURE PART 6.4.2

In	source picture part name	PartName
In	sink picture part name	PartName
In	nameset	NameSet

Primitives in the picture part specified by source picture part name are appended to the picture part specified by sink picture part name. The names in the specified nameset are added to the nameset attributes bound to each of the primitives in the picture part.

RENAME PICTURE PART 6.4.1

In	old picture part name	PartName
In	new picture part name	PartName

The specified picture part is renamed. If old picture part name is the name of the open picture part, the 'name of the open picture part' in the GKS state list is set to new picture part name.
The old picture part name may be reused by the application program.

DELETE PICTURE PART 6.4.2

In	picture part name	PartName

The specified picture part is deleted. The picture part name may be reused by the application program.

COPY PICTURE PART TO NDC PICTURE 6.4.3

In	picture part name	PartName
In	selection criterion	SelectCrit
In	transformation matrix	Matrix23
In	nameset	NameSet

Primitives in the specified picture part in picture part store which satisfy the specified selection criterion are added to the NDC picture after transformation by the specified transformation matrix. The names in the specified nameset are added to the nameset attributes bound to each of the primitives in the picture part. The current sets of clipping rectangles and shielding rectangles are bound to each primitive.

CREATE IMAGE PICTURE PART 6.4.4

In	picture part name		PartName
In	image function		ImageFunction

The specified picture part is defined by the image function.

7.9 Input functions

SET LOGICAL INPUT DEVICE MODE 6.7.1, 6.7.3

In	device identifier	DeviceId
In	operating mode	Mode

The specified logical input device is set to the specified operating mode. Depending on the specified operating mode, an interaction with the given device may begin or end. The input device state defined by 'operating mode' is stored in the workstation state list for the given device.

REQUEST INPUT 6.7.1, 6.7.2, 6.7.3, 6.7.4, 8.10, 8.11

In	device identifier	DeviceId
Out	status	InputStatus
Out	logical input value	InputValue

GKS performs a REQUEST on the specified logical input device. If the break facility is invoked by the operator, the status NONE is returned; otherwise OK is returned together with the logical input value.

SAMPLE INPUT 6.7.1, 6.7.2, 6.7.3, 6.7.4, 8.10, 8.11

In	device identifier	DeviceId
Out	logical input value	InputValue

The current logical input value of the specified logical input device is returned.

AWAIT INPUT 6.7.1, 6.7.2, 6.7.3, 6.7.4, 8.10, 8.11

In	timeout (seconds)	Timeout
Out	device identification and logical input value	P (DeviceId × InputValue)

If the input queue is empty, GKS is set into a wait state until an input event is written into the queue or the time specified in the timeout parameter has elapsed. If a timeout occurs and there is still no entry in the queue, a NONE value is returned for device identification. If there is at least one entry in the queue, the oldest event is returned. The device identifier is returned together with the corresponding logical input value.
The operation is performed even if input queue overflow (error 17) has occurred.

A timeout of zero causes an immediate inspection of the queue, and a NONE value for device identification is returned if the queue is empty.

Some operating systems may not provide a reliable timeout facility. In this case a timeout different from zero may never cause a timeout at all.

FLUSH DEVICE EVENTS 6.7.5

In	device identifier	DeviceId

All entries in the input queue from the specified logical input device are removed. The operation is performed even if input queue overflow (error 17) has occurred.

7.10 Inquiry functions

Inquiry functions return values from the various state lists. The data types of the values and the default values of the state list entries are summarized in clause 10. Errors detected by inquiry functions are reported through an error indicator parameter, see 6.10. The error handling procedure is not called.

INQUIRE OPERATING STATE VALUE 6.10

Out	GKS operating state value	GKSOpSt
Out	picture part store operating state value	PPOpSt

The operating states are returned.

INQUIRE GKS DESCRIPTION TABLE 6.10

Out	error indicator	ErrorIndicator
Out	workstation description table	WDT

If the inquired information is available, the error indicator is returned as 0 and values are returned in the output parameter.

If the inquired information is not available, the values returned in the output parameters are implementation dependent and the error indicator is set to the following error number to indicate the reason for non-availability:

2 GKS not open

INQUIRE GKS STATE LIST 6.10

Out	error indicator	ErrorIndicator
Out	GKS state list	GSL

If the inquired information is available, the error indicator is returned as 0 and values are returned in the output parameter.

If the inquired information is not available, the values returned in the output parameters are implementation dependent and the error indicator is set to the following error number to indicate the reason for non-availability:

2 *GKS not open*

INQUIRE INPUT QUEUE OVERFLOW 6.7.5, 6.10

Out	error indicator	ErrorIndicator
Out	device identifier	DeviceId

If the inquired information is available, the error indicator is returned as 0 and values are returned in the output parameters.

If the input queue has overflowed since OPEN GKS or the last invocation of INQUIRE INPUT QUEUE OVERFLOW, the identification of the logical input device that caused the overflow is returned. The entry is removed from the error state list.

If the inquired information is not available, the values returned in the output parameters are implementation dependent and the error indicator is set to one of the following error numbers to indicate the reason for non-availability:

2 *GKS not open*
35 *Input queue has not overflowed ever or since the last invocation of INQUIRE INPUT QUEUE OVERFLOW*
36 *Input queue has overflowed, but associated workstation has been closed*

8 Workstation dependent control

8.1 Introduction

As shown in figure 2, GKS-N has a set of functions which create and modify the NDC picture. When a workstation is opened, a selection criterion is established for the workstation which selects part of the NDC picture for rendering on the workstation.

When primitives are created, the relevant attributes are bound to the primitive before it is added to the NDC picture or picture part store. The GKS-N state list contains the current values of all the attributes that can be bound to primitives on creation.

The operation of rendering the NDC picture on a workstation proceeds in two stages. In the first stage, unbound attributes of primitives are bound by accessing the relevant workstation dependent tables and the resulting primitives are rendered to produce the logical picture which is still in NDC space. In GKS-N, colour is specified indirectly and in the logical picture colour is not resolved beyond colour indices.

The second stage applies the workstation transformation, workstation clip and replaces colour indices by the colours defined in the colour table for the workstation to produce the physical picture which is displayed on the workstation display space. The physical picture can be sent to a metafile.

The updating of the logical and physical pictures is continuous. Any change to the NDC picture produces a change as soon as possible in the logical picture. The physical picture will have any colour or workstation transformation changes made as soon as possible also.

8.2 Workstation characteristics

For every type of workstation present in a GKS-N implementation there exists a generic workstation description table which describes the standard capabilities and characteristics of the workstation. When the workstation is opened, a new specific workstation description table is created for that workstation containing information which is derived from the following: the generic workstation description table, the device itself, and possibly other implementation dependent sources. The content of the specific workstation description table may change at any time while the workstation is open. The application program can inquire which generic capabilities are available before the workstation is open. The specific capabilities may be inquired while the workstation is open by first inquiring the workstation type of an open workstation, to obtain the workstation type of the specific workstation description table, and then using this workstation type as a parameter to the inquiry functions which query the workstation description table. This information may be used by the application program to adapt its behaviour accordingly. If capabilities are requested that a particular workstation does not provide or, in the case of the generic workstation description table, is not yet available a standard error reaction is defined.

An abstract graphical workstation with maximum capabilities

a) has one addressable display space of fixed resolution;

b) allows only rectangular display spaces (the display space does not consist of a number of separate parts);

c) permits the specification and use of smaller display spaces than the maximum while guaranteeing that no display image is generated outside the specified display space;

d) renders the NDC picture to produce the physical picture using workstation dependent logical attributes;

e) has one or more logical input devices for each input class;

f) permits REQUEST, SAMPLE and EVENT type input;

g) allows logical input devices to be set in REQUEST, SAMPLE or EVENT mode independently of each other.

There are three different types of capability:

h) An explicitly defined and required capability. Every GKS-N implementation supports the capability.

i) An explicitly defined and non-required capability. A GKS-N implementation may support the capability and, if it does, it is implemented according to the explicit function definitions.

j) A conceptually defined and non-required capability. A GKS-N implementation may provide the capability. Its implementation follows general rules given by the GKS-N concepts and functional definitions.

The set of explicitly defined and required capabilities includes:

k) predefined bundle numbers up to the required minimum;

l) linetypes 1 to 4

m) marker types 1 to 5;

n) text precision STROKE;

o) interior style HOLLOW;

p) one input device for each input class;

q) prompt and echo type 1;

r) colour models RGB and CIELUV.

The set of explicitly defined and non-required capabilities include:

s) interior style SOLID, PATTERN, HATCH;

t) transformable patterns;

u) prompt and echo types above 1 that are defined;

The set of conceptually defined and non-required capabilities includes:

v) linetypes above 4;

w) marker types above 5;

x) specific generalized drawing primitives;

y) prompt and echo types above the defined set;

z) specific escape functions.

Table 1 identifies the minimum support required.

Table 1 - Minimum support required

CAPABILITY	Minimum value
Colour models (see note 5)	2
Foreground colours (intensity)	1
Linetypes	4
Linewidths	1
Predefined polyline bundles	5
Settable polyline bundles	20
Marker types	5
Marker sizes	1
Predefined polymarker bundles	5
Settable polymarker bundles	20
Character heights (see note 1)	1
Character expansion factors (see note 1)	1
String precision fonts	1
Character precision fonts	1
Stroke precision fonts	2
Predefined text bundles	6
Settable text bundles	20
Predefined patterns (see note 2)	1
Settable patterns (see notes 2 and 4)	10
Hatch styles (see note 3)	3
Predefined fill area bundles	5
Settable fill area bundles	10
Input classes	6
Prompt and echo types per device	1
Length of input queue (see note 4)	20
Maximum string buffer size (characters)	72
Maximum stroke buffer size (points)	64

NOTES

1 Relevant only for character and string precision text.

2 Relevant only for workstation supporting pattern interior style.

3 Relevant only for workstation supporting hatch interior style.

4 Since available resources are finite and entries have variable size, it may not always be possible to achieve the minimum values in a particular application.

5 The colour models RGB and CIELUV are mandatory.

Actual workstations may provide more capabilities than those listed in the workstation description table. These cannot be used by GKS-N. However, if the workstation itself provides sufficient intelligence, the additional capabilities may be accessed via the GENERALIZED DRAWING PRIMITIVE or ESCAPE functions, or used 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-N or the application program. More than one display space can be controlled by GKS-N only by defining a separate workstation for each display space.

8.3 Selecting a workstation

The application program references a workstation by means of a workstation identifier. Connection to a particular workstation is established by the function OPEN WORKSTATION, which associates the workstation identifier with a workstation type. The current state of each open workstation is kept in a workstation state list.

Connections are released by the function CLOSE WORKSTATION.

8.4 Workstation transformations

The normalized device coordinate space can be regarded as a workstation independent abstract viewing surface. Each open workstation can select independently some part of the NDC space in the range [0,1]×[0,1] to be displayed somewhere in the workstation display space. A particular workstation transformation is a mapping from NDC space onto the device coordinates (DC) for that workstation.

The units of device coordinates are metres on a device capable of producing a precisely scaled image (for example, on most plotters) and appropriate workstation dependent units otherwise (for example, on a display unit with unknown monitor size). In either case the device coordinate system maps onto the display space in the following way:

a) the DC origin is at the bottom left corner of the display space;

b) the device coordinate units are related to the display space in such a way that a square in device coordinates appears as a square on the display surface (this is trivially true if device coordinate units are metres);

c) x and y increase to the right and upwards respectively.

On some devices, device coordinate units do not coincide with addressable units, for example if the addressable units do not satisfy the above conditions.

The current size of the display space in device coordinate units is recorded in the specific workstation description table (see 8.2).

The workstation transformation is a uniform mapping from NDC onto DC and thus performs translation and *equal* scaling with a positive scale factor for the two axes. Thus picture composition can be achieved using the normalization transformations whereas the workstation transformation allows different aspects of the composed picture to be viewed on different workstations. For example, a drawing could be output to a plotter at the correct scale and simultaneously some part of the drawing could be displayed on the full display surface of an interactive terminal.

The workstation transformation can be specified at any time after the workstation has been opened.

A workstation transformation is specified by defining the limits of an area in the normalized device coordinate system within the range [0,1]×[0,1] (workstation window) which is to be mapped onto a specified area of the device coordinate space (workstation viewport). Workstation window and workstation viewport limits specify rectangles parallel to the coordinate axes in NDC and DC. The rectangles include their boundaries. To ensure that no output outside the workstation window is displayed, GKS clips at the workstation window boundaries, and this clipping cannot be disabled. As the workstation window is defined somewhere in the NDC range [0,1]×[0,1], this ensures that the only part of NDC space that can be viewed on any workstation lies in the range [0,1]×[0,1]. If the workstation window and workstation viewport have different aspect ratios, the scaling specified would be different on each axis if the workstation window was mapped onto the workstation viewport in its entirety. To ensure equal scaling on each axis, the workstation transformation maps the workstation window onto the largest rectangle that can fit within the workstation viewport such that:

d) aspect ratio is preserved;

e) the lower left-hand corner of the workstation window is mapped to the lower left-hand corner of the workstation viewport.

Thus, space is left unused at the top or right side of the workstation viewport if the aspect ratios of the workstation window and workstation viewport are different.

All workstation transformations are set by default to map NDC space [0,1]×[0,1] onto the whole of the workstation display space. If the display space is not square, the same rules as above apply to achieve equal scaling on each axis.

Workstation transformations are changed by the SET WORKSTATION WINDOW AND VIEWPORT function.

8.5 Geometric primitives

All attributes are bound to geometric primitives on creation. The physical picture is then generated by looking up the workstation's colour table where appropriate and applying the workstation transformation and clip.

8.6 Bundled primitives

Each bundled primitive has four types of attributes (NDC, logical, physical and identification). NDC attributes are bound to the bundled primitive on creation. Logical attributes can either be bound on creation or bound when the logical picture is generated from the NDC picture. Identification attributes are bound on creation (although the nameset attribute can be altered later) and identify what part of the NDC picture is rendered on a workstation (using the selection criterion) and are used to identify the primitives that input values are associated with.

The four bundled primitives, POLYLINE, POLYMARKER, FILL AREA and TEXT, each have a set of logical attributes which can either be bound to the NDC picture or at the workstation when the NDC picture is being rendered to produce the logical picture. The sets of attributes are:

Primitive	Logical attributes
POLYLINE	LINETYPE LINEWIDTH SCALE FACTOR POLYLINE COLOUR INDEX
POLYMARKER	MARKER TYPE MARKERSIZE SCALE FACTOR POLYMARKER COLOUR INDEX
TEXT	TEXT FONT AND PRECISION CHARACTER EXPANSION FACTOR CHARACTER SPACING TEXT COLOUR INDEX
FILL AREA	FILL AREA INTERIOR STYLE FILL AREA STYLE INDEX FILL AREA COLOUR INDEX

For each class of primitive, a bundle table exists on each workstation. Each bundle table entry contains values for the set of logical attributes associated with that class. For example, the workstation POLYLINE bundle table will have a set of entries each of which will contain values for LINETYPE, LINEWIDTH SCALE FACTOR and POLYLINE COLOUR INDEX. Values for each of these attributes are also present in the GKS state list. The GKS state list also contains entries for an index value for each bundled primitive class and a set of ATTRIBUTE SOURCE FLAGS (ASFs), one for each attribute that can either be bound in the NDC picture or at the workstation when the logical picture is created. The initial values of all the ASFs are the same. It is implementation dependent whether they are BUNDLED or INDIVIDUAL.

On the creation of a bundled primitive, the following attributes are bound:

a) all NDC and identification attributes;

b) all logical attributes of the primitive where the corresponding ASF is set to INDIVIDUAL (rather than BUNDLED);

c) If some logical attributes are not bound on creation, the bundled primitive index is bound to the primitive.

At the workstation, the logical picture is created as follows for each bundled primitive that satisfies the selection criterion:

d) no further binding if all logical attributes are already bound;

e) if the bundled primitive index is bound to the primitive, this indicates that some logical attributes have not yet been bound. The entry in the bundle table for that index value will contain the attribute values still to be bound. The bundled primitive index does not appear in the attributes of the primitives in the logical picture.

Figure 12 shows, for the POLYLINE primitive, the effect for different settings of the ASFs. Names have been used in place of values for some attributes to make the figure clearer.

The bundled primitive CELL ARRAY has all its attributes bound on creation. These include an array of colour indices. Consequently, if the primitive appears in the logical picture of a workstation, the attributes bound to it are the same in both the logical and NDC pictures. The physical picture is obtained by looking up all the colour indices in the workstation's colour table and replacing them by the colours specified in the table (see 8.7.6).

Mapping the transformed cells onto the pixels of a raster display (see figure 11) is performed by the following rules:

f) If the centrepoint of a pixel lies inside the parallelogram defined by the transformed rectangle, its colour is set.

g) The pixel is assigned the colour of the cell which contains the pixel's centrepoint. Thus, the pixel colour is selected by point sampling the transformed rectangle at the pixel centrepoint, not by area sampling or filtering.

The minimal simulation required is to draw the transformed boundaries of the cell rectangle, using implementation dependent colour, linewidth and linetype.

The bundled primitive GDP uses the attributes of other bundled primitive classes. If used by a GDP, the attributes of the other primitives behave in the same way as for the primitive class itself.

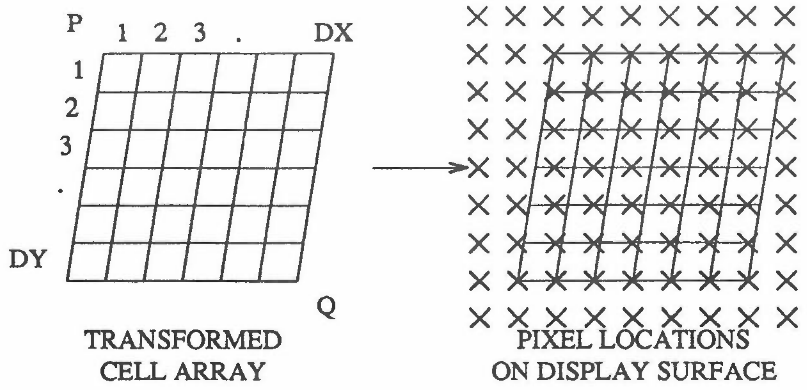


Figure 11 - Mapping CELL ARRAY cells onto pixels

8.7 Bundled primitive attributes

8.7.1 Introduction

Some logical attributes have a range of values (say 1 to 5). For attributes of that type, values greater than the maximum defined are reserved for registration. Values less than 0 may be available but their meanings are implementation dependent. The value 0 is not allowed.

The colour index for each primitive defines an entry in the workstation colour table.

The following subclauses describe the logical attributes of each bundled primitive class.

8.7.2 Polyline logical attributes

The polyline logical attributes are LINETYPE, LINEWIDTH SCALE FACTOR and POLYLINE COLOUR INDEX.

The defined linetypes 1 to 4 are solid, dashed, dotted, and dashed-dotted. The linetype specifies a sequence of line segments and gaps which are repeated to define the rendering of the polyline. It is workstation dependent whether this sequence is restarted or continued at the start of the polyline, at the start of a clipped piece of a polyline, and at each vertex of a polyline.

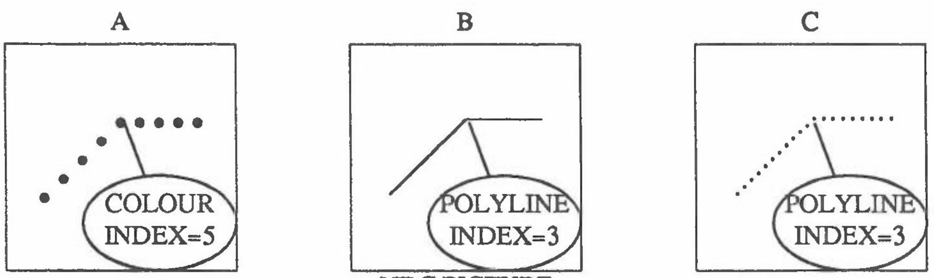
The linewidth is calculated as a nominal linewidth multiplied by the linewidth scale factor. This value is mapped by the workstation to the nearest available linewidth.

8.7.3 Polymarker logical attributes

The polymarker logical attributes are MARKER TYPE, MARKERSIZE SCALE FACTOR, and POLYMARKER COLOUR INDEX.

Marker types 1 to 5 are dot, plus sign, asterisk, circle and diagonal cross each centred on the positions they are identifying.

GKS STATE LIST			
	A	B	C
POLYLINE INDEX	3	3	3
LINETYPE	Dotted	Dotted	Dotted
LINEWIDTH SCALE FACTOR	Thick	Thick	Thick
POLYLINE COLOUR INDEX	5	5	5
ASFs			
LINETYPE	INDIVIDUAL	BUNDLED	INDIVIDUAL
LINEWIDTH SCALE FACTOR	INDIVIDUAL	BUNDLED	BUNDLED
POLYLINE COLOUR INDEX	INDIVIDUAL	BUNDLED	BUNDLED



NDC PICTURE

WORKSTATION

POLYLINE BUNDLE TABLE

	LINETYPE	LINEWIDTH SCALE FACTOR	POLYLINE COLOUR INDEX
1	SOLID	NORMAL	1
2	DOTTED	THICK	4
3	DASHED	NORMAL	3
4	SOLID	THIN	2

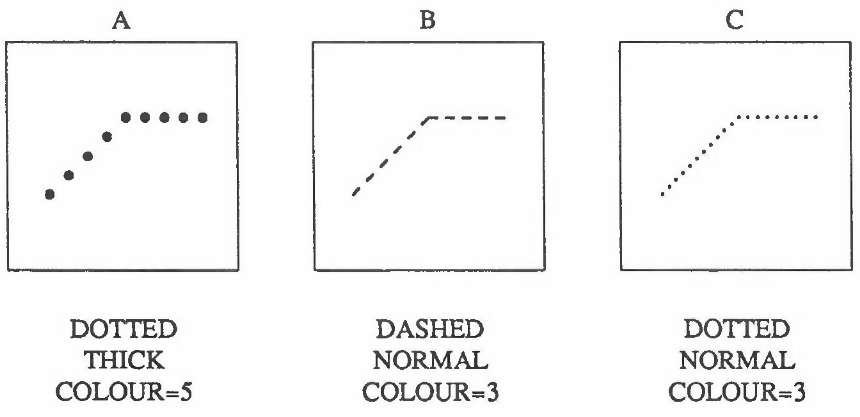


Figure 12 - Effect of different ASF settings

The marker size is calculated as a nominal size multiplied by the markersize scale factor. This size is mapped by the workstation to the nearest available size. Marker type 1 is always displayed as the smallest displayable dot.

The marker is visible if, and only if, the marker position is within the clipping rectangle. The clipping of partially visible markers is workstation dependent.

8.7.4 Text logical attributes

Text has the NDC attributes CHARACTER HEIGHT, CHARACTER UP VECTOR, TEXT PATH and TEXT ALIGNMENT. The first three have been described in 6.2.10.

The logical attributes are TEXT FONT AND PRECISION, CHARACTER EXPANSION FACTOR, CHARACTER SPACING and TEXT COLOUR INDEX.

The CHARACTER SPACING value specifies how much additional space is to be inserted between two adjacent character bodies. If the value of CHARACTER SPACING is zero, the character bodies are arranged one after the other along the TEXT PATH, without any additional space between. A positive value of CHARACTER SPACING will insert additional space between character bodies. A negative value of CHARACTER SPACING will cause adjacent character bodies to overlap. CHARACTER SPACING is specified as a fraction of the font nominal character height.

The effect of the attributes CHARACTER HEIGHT, TEXT PATH, CHARACTER EXPANSION FACTOR, CHARACTER SPACING and TEXT FONT AND PRECISION is to define an (imaginary) rectangle with its sides parallel to the x and y axes, enclosing the text. The bounds of this enclosing rectangle are as follows. For TEXT PATH = LEFT or RIGHT, the height of the rectangle is the height of the character body of the specified font; the left side of the rectangle is the left side of the character body of the leftmost character and the right side of the rectangle is the right side of the character body of the rightmost character. For TEXT PATH = UP or DOWN, the top of the rectangle is the top of the character body of the topmost character and similarly, the bottom of the rectangle is the bottom of the bottommost character; the width of the rectangle is the width of the widest character in the specified font.

The effect of the CHARACTER UP VECTOR attribute is to transform the enclosing rectangle, thus defining an enclosing parallelogram, the text extent parallelogram (the rectangle has been rotated and sheared).

The TEXT ALIGNMENT attribute controls the positioning of text in relation to the text position. For simplicity the TEXT ALIGNMENT is described in terms of the default CHARACTER UP VECTOR and implied baseline direction, when the text extent parallelogram is actually a rectangle. The horizontal component of TEXT ALIGNMENT has four values: LEFT, CENTRE, RIGHT and NORMAL. If the horizontal component is LEFT, the left side of the text extent parallelogram passes through the text position. Similarly, if the value is RIGHT, the right side of the text extent parallelogram passes through the text position. If the horizontal component is CENTRE, the text position lies midway between the left and right sides of the text extent parallelogram. Thus, if TEXT PATH = UP or DOWN, the straight line passing through the centrelines of the characters also passes through the text position. The vertical component of TEXT ALIGNMENT has six values: TOP, CAP, HALF, BASE, BOTTOM and NORMAL. These each correspond to one of the font specific horizontal lines in the definition of a character (see figure 1). A value of TOP causes the top of the text extent parallelogram to pass through the text position. A value of CAP causes the text position to lie on the capline of the whole string (TEXT PATH = LEFT or RIGHT) or on the capline of the topmost character in the string (TEXT PATH = UP or DOWN). A value of HALF causes the text position to lie on the halfline of the whole string (TEXT PATH = LEFT or RIGHT) or on a line halfway between the halflines of the top and bottom characters (TEXT PATH = UP or DOWN). A value of BASE causes the text position to lie on the baseline of the whole string (TEXT PATH = LEFT or RIGHT) or on the baseline of the bottom character in the string (TEXT PATH = UP or DOWN). A value of BOTTOM causes the bottom of the text extent parallelogram to pass through the text position.

In the general case, the orientation referred to as horizontal is that of the implied baseline direction with RIGHT representing direction of that vector and LEFT being opposite to it. Similarly the orientation referred to as vertical is that of the CHARACTER UP VECTOR with UP representing the direction of that vector and DOWN being opposite to it.

Either component of TEXT ALIGNMENT can take the value NORMAL. For each value of TEXT PATH, the effect of a particular component being NORMAL is equivalent to one of the other values of that component. In each case, the equivalent alignment value is chosen to achieve a natural alignment for that TEXT PATH value. The complete list of equivalent values is:

TEXT PATH	NORMAL Horizontal and Vertical Alignments
RIGHT	(LEFT, BASE)
LEFT	(RIGHT, BASE)
UP	(CENTRE, BASE)
DOWN	(CENTRE, TOP)

Text font and precision together constitute one attribute. The text font value is used to select a particular font on the workstation. Every workstation supports at least one font that is able to generate a graphical representation of the characters defined in ISO 646. This is font number 1. Font numbers greater than 1 are reserved for registration. Font numbers less than 0 may be supported but are implementation dependent.

The text precision value is used to select the 'closeness' of the text representation at the workstation in relation to that defined by the NDC text attributes and the transformation and clipping currently applicable. The text precision value has the following possible values:

a) STRING: The TEXT character string is generated in the requested text font and is positioned by aligning the TEXT output primitive at the given text position. CHARACTER HEIGHT and CHARACTER EXPANSION FACTOR are evaluated as closely as reasonable, given the capabilities of the workstation. CHARACTER UP VECTOR, TEXT PATH, TEXT ALIGNMENT and CHARACTER SPACING, need not be used. Clipping is done in an implementation and workstation dependent way.

b) CHAR: The TEXT character string is generated in the requested text font. For the representation of each individual character, the attributes CHARACTER HEIGHT, the implied width, the up direction of the CHARACTER UP VECTOR, the baseline direction, and CHARACTER EXPANSION FACTOR are evaluated as closely as possible, in a workstation dependent way. The spacing used between character bodies is evaluated exactly; the character body, for this purpose, is an ideal character body, calculated precisely from the text attributes and the font dimensions. The position of the resulting text extent parallelogram is determined by the TEXT ALIGNMENT and the text position. Clipping is performed at least on a character by character basis.

c) STROKE: The text character string in the requested text font is displayed at the text position by applying all text attributes. The character string is clipped exactly at the clipping rectangle.

STROKE precision does not necessarily mean vector strokes; as long as the representation adheres to the rules governing STROKE precision, the font may be realized in any form, for example by raster fonts.

All text precisions are supported as follows. A workstation may use a higher precision than the one requested for this purpose i.e. if STROKE precision is supported in a particular font, the implication is that both STRING and CHAR precision are available in that font. However it is not necessary for a workstation to support all precisions for a given font (i.e. for a given font, STROKE can be missing or both STROKE and CHAR can be missing). Text font and precision are workstation mandatory. Every workstation of a particular installation should support at least one STROKE precision text font. This is font number 1, containing the character set defined by ISO 646. This implies that, for STROKE precision text, some sort of software character generator is required for those implementations that have inadequate hardware. Not all workstations need to support all fonts, but for those that do, the same font number is used to select that font on all workstations of a particular installation.

Fonts are defined only within the GKS-N implementation. The font designer specifies the shape of the symbol representing each character in a local 2D cartesian font coordinate system. Fonts are either monospaced or proportionally spaced. Each character in a font coordinate system has an associated character body, a font baseline, a font halfline, a capline and a centreline (see figure 3). For monospaced fonts, the character bodies of all characters have the same size. For proportionally spaced fonts, the width of the bodies may differ from character to character. The character body edges are parallel to the axes of the font coordinate system. The font baseline, the font halfline and the capline are parallel to the x-axis of the font coordinate system, and within the vertical extent of the body. The position of the font halfline is defined by the font designer for use in aligning text strings. The centreline is parallel to the y-axis and bisects the body. Their exact positions are specified by the font designer.

The height of a character in the font coordinate system is given by the height from the font baseline to the capline. The width of a character is given by the width of the character body. The width of a character may include space on either side of the character and this space is generally evenly split between the left and right sides of the character. It is assumed that the characters lie within their body, except that kerned characters may exceed the side limits of the character body.

In general, the top limits of the bodies for a font are identical with, or very close to, the typographical capline or ascender line, and the bottom limit to the descender line. The space, if any, between the topline and the capline may be used for an additional mark over the character, for example an accent. However, these and other details are purely for the use of the font designer. The intention is only that characters placed with their bodies touching in the horizontal direction should give an appearance of good normal spacing, and characters touching in the vertical direction will avoid clashes between ascenders and descenders (typographically 'set solid').

Since the values of CHARACTER HEIGHT and CHARACTER UP VECTOR are given in world coordinate units, but the characters are generated on the workstation in device coordinates, using the workstation dependent font and precision, the NDC attributes need to be transformed in such a way that the workstation can generate the characters in the way intended.

The effect to be achieved is now described. Together with the text coding, a height vector parallel to the CHARACTER UP VECTOR with length equal to CHARACTER HEIGHT, and a width vector parallel to the implied baseline direction with length equal to the implied character width, are passed down the viewing pipeline. These vectors are transformed by the normalization transformation, by a picture part transformation if within a picture part, and by the workstation transformation. They are also stored in picture parts. Then the vectors can be used by the workstation character generator. Thus, the shape of individual characters can be transformed by a normalization transformation that is unequal in x and y and by a picture part transformation.

On the workstation, the height of a character is given by the length of the transformed height vector; the character up direction is given by the direction of the transformed height vector; the width of a character is given by the length of the transformed width vector multiplied by the font width to height ratio for the character and by the CHARACTER EXPANSION FACTOR; the character base direction is given by the direction of the transformed width vector. The characters are arranged together in a text extent parallelogram, depending on the values of TEXT PATH and CHARACTER SPACING. The text extent parallelogram is then positioned according to the value of TEXT ALIGNMENT and the text position, contained in the definition of the TEXT primitive.

Figures 13 gives examples of the effects of different values of text attributes. Figure 14 gives examples of the effect of different normalization transformations on the displayed form of the text. Figure 15 shows the effect of a set of attributes being specified.

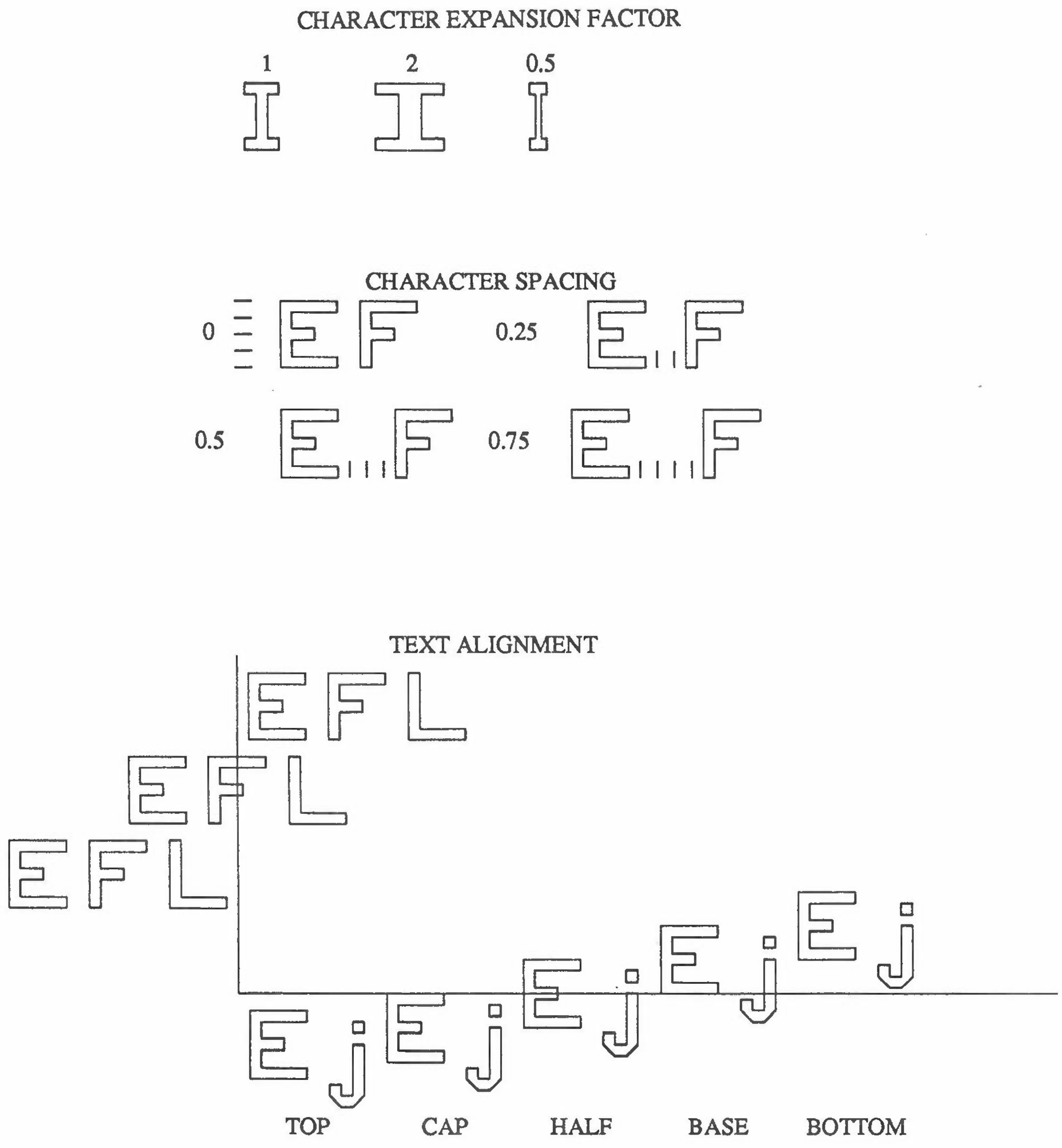


Figure 13 - Text attributes

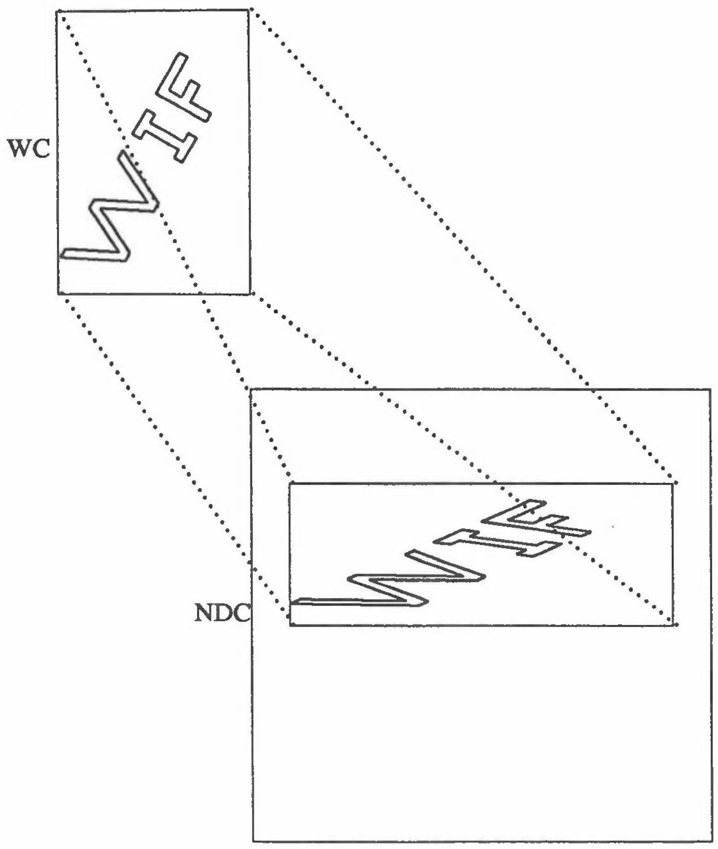


Figure 14 - Effect of normalization transformation on text

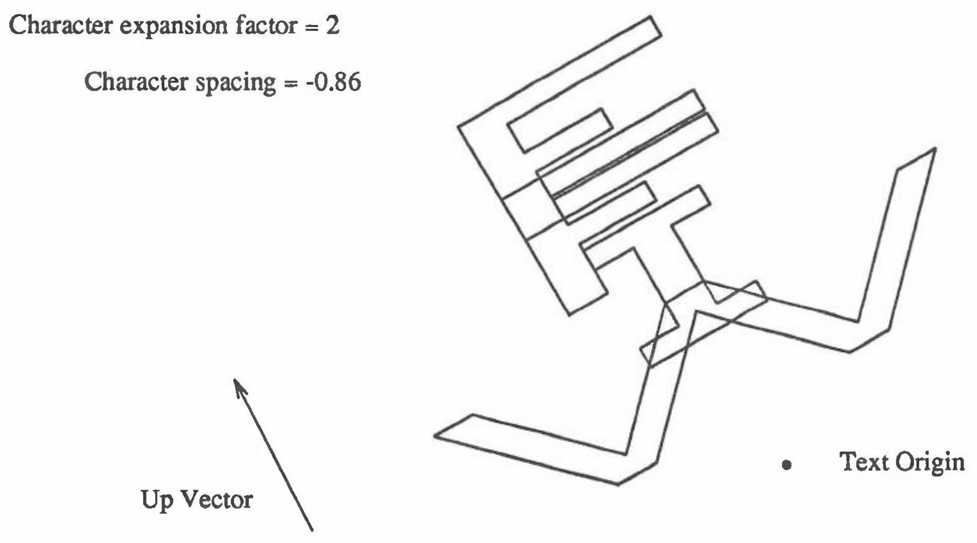


Figure 15 - Effect of several text attributes together

GKS-N provides a function INQUIRE TEXT EXTENT which returns an estimate of the area a given text string will occupy when displayed with the current NDC and logical attributes on a specified workstation, together with a concatenation point which can be used as the origin of a subsequent TEXT output primitive for the concatenation of character strings, where this is meaningful.

At precisions STRING and CHAR, the text extent parallelogram is an approximation of that defined above, being the minimum which completely encloses the character bodies of the displayed string. For UP and DOWN text paths, the widest character body in the font is enclosed. The parallelogram is returned as four corner points in anticlockwise order. If, at STROKE precision, the CHARACTER WIDTH VECTOR and CHARACTER BASE VECTOR are perpendicular, the text extent parallelogram is a rectangle.

The concatenation point can be used as the origin of a subsequent TEXT output primitive for the concatenation of character strings, where meaningful. For certain combinations of TEXT PATH and TEXT ALIGNMENT, concatenation is not meaningful and the returned concatenation point is the same as the text position.

If TEXT PATH is RIGHT or LEFT, the concatenation point is displaced from the text position, in a direction determined by the horizontal component of TEXT ALIGNMENT. If this component is LEFT, the direction is to the right; if it is CENTRE, the displacement is zero; if it is RIGHT, the direction is to the left. Unless the horizontal component of TEXT ALIGNMENT is CENTRE, the magnitude of the displacement is the width of the text extent parallelogram plus one additional character spacing. (The width of the text extent parallelogram is the length of the sides parallel to the CHARACTER BASE VECTOR.)

If TEXT PATH is UP or DOWN, the concatenation point is displaced from the text position in a direction determined by the vertical component of TEXT ALIGNMENT. If this component is TOP or CAP, the direction is down; if it is HALF, the displacement is zero; if it is BASE or BOTTOM, the direction is up. Unless the vertical component of TEXT ALIGNMENT is HALF, the magnitude of the displacement is the height of the text extent parallelogram plus an additional character spacing. (The height of the text extent parallelogram is the length of the sides parallel to the CHARACTER UP VECTOR.)

8.7.5 Fill area logical attributes

Fill area has the NDC attributes PATTERN REFERENCE POINT and SET PATTERN SIZE. If the PATTERN SIZE is (SX,SY), this defines the pattern height vector as (0,SY) and pattern width vector as (SX,0). These values are subject to the same transformations as the geometric data contained in the definition of the primitive. The usage of the fill area NDC attributes is described later in this sub-clause.

The logical attributes of fill area are FILL AREA INTERIOR STYLE, FILL AREA STYLE INDEX and FILL AREA COLOUR INDEX.

The FILL AREA INTERIOR STYLE is used to determine in what style the area should be filled. It has the following values:

a) HOLLOW:	No filling, but draw the bounding polyline, using the FILL AREA COLOUR INDEX currently selected. The linetype and linewidth are implementation dependent.
b) SOLID:	Fill the interior of the polygon using the FILL AREA COLOUR INDEX currently selected.
c) PATTERN:	Fill the interior of the polygon using the FILL AREA STYLE INDEX currently selected as an index into the pattern table. In this context, the FILL AREA STYLE INDEX is sometimes referred to as the pattern index.
d) HATCH:	Fill the interior of the polygon using the FILL AREA COLOUR INDEX and the FILL AREA STYLE INDEX currently selected. The FILL AREA STYLE INDEX is used as a pointer into the list of hatch styles, in which case it is sometimes referred to as the hatch index.

For interior style PATTERN, the pattern is defined by the pattern representation, which specifies an array (DX×DY) of colour indices, that are pointers into the colour table. The size and position of the start of the pattern are determined by a pattern box. The pattern box, which is a parallelogram, is defined by the pattern width vector and the pattern height vector located relative to the PATTERN REFERENCE POINT. The pattern box is conceptually divided into a grid of DX×DY cells. The colour index array is associated with the cells as follows: the element (1,DY) is associated with the cell having the PATTERN REFERENCE POINT at one corner; elements with increasing first dimension are associated with successive cells in the direction of the pattern width vector; elements with decreasing second dimension are associated with successive cells in the direction of the pattern height vector. The attributes defining the pattern box are subject to all the transformations producing a transformed pattern box. The pattern is mapped onto the polygon by conceptually replicating the transformed pattern box in directions parallel to its sides until the interior of the complete polygon is covered.

Mapping the transformed pattern cells to the pixels of a raster display is performed by the following rules:

e) if the centre of a pixel lies inside the parallelogram defined by the transformed cell, its colour is set;

f) the pixel is assigned the colour of the cell corresponding to the pixel's centre.

For a workstation which can implement patterns but not transformable patterns, a suitable action is to generate non-transformed patterns to fill a polygon.

For interior style HATCH, the hatch index selects among hatch styles. No hatch styles are predefined. Whether hatching is affected by transformations or not is workstation dependent.

Interior style HOLLOW is available on every workstation. It is workstation dependent which of the interior styles SOLID, PATTERN and HATCH are available.

8.7.6 Colour

In GKS-N, colour is specified in a number of different situations. It may be an attribute of a primitive, in which case it is specified as a colour index. It may be part of a pattern for FILL AREA, in which case an array of colour indices is specified, or it may be part of a primitive itself, namely CELL ARRAY, when an array of colour indices is also specified. In each case, the colour is specified as an index into a colour table on the workstation. On each workstation, there is one colour table into which all the colour indices point.

The size of the colour table is workstation dependent but entries 0 and 1 always exist. Entry 0 corresponds to the background colour. The background colour is the colour of the display space after it has been cleared. Entry 1 is the default foreground colour and entries higher than 1 correspond to alternative foreground colours. Colours are described by a colour model together with a specification of colour coordinates in the colour space of that model. There is a colour model in effect at each workstation, specified by the 'current colour model entry' in the workstation state list.

Colours are associated with a colour index by the function SET COLOUR REPRESENTATION. The function INQUIRE WORKSTATION STATE LIST can be used to determine the colour associated with a colour index. The parameters of these functions describe the colour as a specification of colour coordinates in the colour space of the current colour model. The number of coordinates necessary to specify colour, and their interpretation, depends on the current colour model.

The current colour model at a workstation is set by the control function SET COLOUR MODEL. This function selects the colour model used for interpretation of the parameters of SET COLOUR REPRESENTATION and INQUIRE WORKSTATION STATE LIST.

Colour models 1 and 2 are predefined as RGB and CIELUV. Colour models greater than 2 are reserved for registration. Colour models less than 1 are implementation dependent. These colour models are described in Annex H.

When colours are associated with a colour index they are mapped to the nearest available on the workstation. On some workstations it may not be possible to change the background colour, and in this case the mapping of a specific colour to the nearest available for the background colour may be different from the mapping of the same colour for the foreground colours.

Some workstations are not capable of displaying colours (for example, workstations only capable of displaying colours with equal red, green, and blue intensities or workstations only capable of displaying colours which are different intensities of the same colour); these are referred to as monochrome workstations. Whether a workstation is capable of colour is recorded in the 'colour available' entry in the workstation description table. On monochrome workstations, the intensity is computed from the colour values in a workstation dependent way.

8.8 Setting primitive attributes

The function SET PRIMITIVE ATTRIBUTE defines all the attribute values in the GKS-N state list that can be bound to a primitive on creation. One value is set at a time by specifying the attribute and the new attribute value to be associated with it. This value is bound to all subsequent primitives to which it refers until the value is reset by another invocation of SET PRIMITIVE ATTRIBUTE for this specific attribute.

The function SET REPRESENTATION defines an entry in a bundle table, the pattern table or the colour table of a workstation. Some standard definitions for data entries are contained in the workstation description table and are used as initial values. The application program may select a standard definition or may define the value of a specific entry explicitly. Only the most commonly used (or anticipated) combinations of values need be predefined for each workstation. At least these predefined entries with indices up to the minimum number of predefined entries are distinguishable from each other. Other combinations of values can be set by the SET REPRESENTATION function, possibly after inquiring the workstation capabilities. The tables which are on every workstation are:

polyline bundle table
polymarker bundle table
text bundle table
fill area bundle table
pattern table
colour table

The values in these tables may be changed at any time. However, they only effect the rendering of primitives added to the NDC picture after the values were changed.

8.9 Selection criterion

The selection criterion for a workstation can be set by invoking the function SET WORKSTATION SELECTION CRITERION. As soon as the selection criterion has been changed, the sequence of primitives in the NDC picture are examined and those that satisfy the new selection criterion have their remaining attributes bound to produce a new physical picture for display or storage.

8.10 Transformation of LOCATOR and STROKE input

The application programmer requires LOCATOR input to define a position in the most appropriate world coordinate system currently defined by the set of normalization transformations.

This is achieved by first transforming the input data from DC to NDC by the inverse workstation transformation which is in effect when LOCATOR input is generated. LOCATOR input can only be obtained from positions within the part of the current workstation viewport into which the current workstation window is mapped (note that this is a subset of the workstation viewport whenever the aspect ratio of the workstation viewport and workstation window differ). Thus, LOCATOR input always defines a position in the NDC range $[0,1]\times[0,1]$.

The mapping from NDC positions to WC positions is described in 6.7.6.

Similar considerations apply to transformation of STROKE input as apply to LOCATOR input, with the complication that more than one point is involved.

When each point of a stroke is generated, the coordinates of the point are transformed from DC to NDC by the inverse workstation transformation then in effect. STROKE input can only be obtained from positions within the part of the current workstation viewport into which the current workstation window is mapped (analogous to LOCATOR input). Thus STROKE input always consists of points in the NDC range $[0,1]\times[0,1]$. The mapping from NDC positions to WC positions is described in 6.7.6.

8.11 Physical metafile

The contents of the physical picture may be captured and stored for future use or transmission to another system. The function COPY PHYSICAL PICTURE TO PHYSICAL METAFILE will store the physical picture as a picture on the specified metafile. Blank pictures may be inserted in the physical metafile by invoking the function COPY BLANK PHYSICAL PICTURE TO PHYSICAL METAFILE. The picture can be recovered at a later time and added to the current physical picture by invoking COPY PHYSICAL METAFILE PICTURE TO PHYSICAL PICTURE.

8.12 Logical input devices

8.12.1 Introduction

A workstation may have one or more logical input devices for each input class. A logical input device is identified by a logical input device number which consists of a workstation identifier, a device class and the device number within that class on that particular workstation.

The application program has some degree of control over logical input devices in that it may supply an initial value for the device, set an area of the display space for the display of prompt and echos and select prompting and echoing techniques to be used.

The application program may also define the composition of compound logical input devices in terms of the measures and triggers provided by the workstation.

8.12.2 Initialization of logical input devices

There is an INITIALIZE LOGICAL INPUT DEVICE function which can only be called if the logical input device it specifies is in REQUEST mode. This function provides the following information to a device via the workstation state list (if the INITIALIZE function is not called, then default values apply):

a) An initial value appropriate to the device. If the initial value violates the rules, an error occurs and the workstation state list is unchanged.

b) A prompt and echo type that selects the prompting technique and, if echoing is on, the echoing technique for a logical input device. An implementation dependent prompt and echo type (type 1) is required for all logical input devices. Further prompt and echo types are defined but not required. Prompt and echo types above those are reserved for registration (see 4.2). Prompt and echo types less than 0 are device dependent.

c) An echo area (xmin,xmax,ymin,ymax) in device coordinates. Input device implementations may use the echo area for certain prompt and echo types to display prompts or echoes.

d) A data record. Some input classes have mandatory control values in the data record. Some prompt and echo types within an input class also have mandatory control values in the data record. These values occupy well defined places in the data record. In any data record used in initializing an input device, values mandatory to the input class, if any, appear first followed by values mandatory to the prompt and echo type if any. Depending on the device and prompt and echo type, the data record may contain other (additional) information.

When a logical input device is REQUESTed, or when it is set to EVENT or SAMPLE mode, its measure is set to the initial value from the workstation state list, unless this is not a valid measure for the device. If it is not a valid measure for the device, the measure is set to a device dependent value, except for PICK devices, for which the measure is set to NOPICK.

Prompt and echo types describe both the prompt, which informs the operator that the device is available, and the echo, which informs the operator of the state of the measure. The functions provided to control input device mode, SET LOGICAL INPUT DEVICE MODE, also control whether echo is on or off. In addition, an implementation dependent acknowledgement of successful trigger firings is provided.

Prompt and echo types for LOCATOR logical input devices are:

<0	prompting and echoing is LOCATOR device dependent.
1	designate the current position of the LOCATOR using an implementation-defined technique.
2	crosshair, i.e. designate the current position of the LOCATOR using a vertical line and a horizontal line spanning over the display surface or the workstation viewport intersecting at the current locator position.
3	designate the current position of the LOCATOR using a tracking cross.
4	designate the current position of the LOCATOR using a rubber band line connecting the initial locator position given by this function and the current locator position.
5	designate the current position of the LOCATOR using a rectangle. The diagonal of the rectangle is the line connecting the initial locator position given by this function and the current locator position.
6	display a digital representation of the current position of the LOCATOR in LOCATOR device dependent coordinates within the echo area.

≥7	reserved for registration.

For some LOCATOR prompt and echo types, two positions are required. One of the positions, which remains fixed during the input operation, is the initial locator position. The other position is the current locator position that varies dynamically as the operator uses the LOCATOR.

Prompt and echo types for STROKE logical input devices are:

<0	prompting and echoing is STROKE device dependent.
1	display the current stroke using an implementation defined technique.
2	display a digital representation of the current stroke position within the echo area.
3	display a marker at each point of the current stroke.
4	display a line joining successive points of the current stroke.
≥5	reserved for registration.

If the operator enters more points than the current input buffer size, the additional points are lost.

Stroke data record entries for variables such as intervals in X, Y and time may be provided to constrain the number of points delivered.

For all prompt and echo types, the first entry in the stroke data record is the input buffer size which is an integer in the range (1..n). This is compared against an implementation defined 'maximum input buffer size' for this device (contained in the workstation description table). If the requested buffer size is greater, the 'maximum input buffer size' is substituted in the stored data record. If the initial stroke specified in the initial value is longer than the buffer size, an error is issued.

When a STROKE measure process comes into existence, it obtains a buffer of the current input buffer size. The initial stroke is copied into the buffer, and the editing position is placed at the initial buffer editing position within it. Replacement of points begins at this initial position. If the initial buffer editing position cannot be specified in the stroke data record, the value 1 is used.

Prompt and echo types for VALUATOR logical input devices are:

<0	prompting and echoing is VALUATOR device dependent.
1	designate the current VALUATOR value using an implementation defined technique.
2	display a graphical representation of the current VALUATOR value within the echo area (for example, a dial or a pointer).
3	display a digital representation of the current VALUATOR value within the echo area.
≥4	reserved for registration.

For all VALUATOR prompt and echo types, the valuator data record includes, in the first two positions, a low value and a high value, in that order, specifying the range. The values from the device will be scaled linearly to the specified range.

Prompt and echo types for CHOICE logical input devices are:

<0	prompting and echoing is CHOICE device dependent.
1	designate the current CHOICE number using an implementation defined technique.
2	the physical input devices that are most commonly used to implement a CHOICE logical input device normally have a built-in prompting capability. This prompt and echo type allows the application program to invoke this prompting capability. If the value of the i-th element of 'prompt array' in the choice data record is OFF, prompting of the i-th alternative of the specified choice input device is turned off. An ON value indicates that prompting for that alternative is turned on. The first entry in the choice data record is the number of choice alternatives. This is compared against an implementation defined 'maximum number of choice alternatives' for this device (contained in the workstation description table). If the maximum value is exceeded, an error is issued. The second entry in the choice data record is the 'prompt array'.

3 allow the operator to indicate a CHOICE number by selecting, using an appropriate technique, one of a set of CHOICE strings. The CHOICE strings are contained in the choice data record and are displayed within the echo area. The logical input value is the number of the string selected. The first entry in the choice data record is the number of choice strings. This is compared against an implementation defined 'maximum number of choice alternatives' for this device (contained in the workstation description table). If the maximum value is exceeded, an error is issued. The second entry in the choice data record is the 'array of choice strings'.

4 allow the operator to indicate a CHOICE number by selecting, via an alphanumeric keyboard, one of a set of CHOICE strings. The CHOICE strings are contained in the choice data record and may be displayed in the echo area as a prompt. The string typed in by the operator is echoed in the echo area. The logical input value is the number of the string that has been typed in by the operator. The first entry in the choice data record is the number of choice strings. This is compared against an implementation defined 'maximum number of choice alternatives' for this device (contained in the workstation description table). If the maximum value is exceeded, an error is issued. The second entry in the choice data record is the 'array of choice strings'.

5 the picture part named by the choice data record is interpreted during execution of the INITIALIZE LOGICAL INPUT DEVICE function for later use as a prompt of the specified CHOICE device. It will be displayed within the echo area by mapping the unit square [0,1]×[0,1] of NDC space onto the echo area. The PICK IDENTIFIERS in the picture part are mapped to CHOICE numbers in a CHOICE device dependent fashion. Picking these primitives selects the corresponding CHOICE value. After the interpretation, no logical connection between the specified picture part and the specified CHOICE device exists. The first entry in the choice data record is the picture part name.

≥6 reserved for registration.

Prompt and echo types for PICK logical input devices are:

<0 prompting and echoing is PICK device dependent.

1 use an implementation-defined technique that at least highlights the 'picked' primitive for a short period of time.

2 echo the contiguous group of primitives within the NDC picture with the same PICK IDENTIFIER and NAMESET as the 'picked' primitive, or all primitives of the NDC picture with the same PICK IDENTIFIER and NAME SET as the 'picked' primitive.

3 echo all the primitives in the NDC picture with the same NAMESET as the 'picked' primitive.

≥4 reserved for registration.

Prompt and echo types for STRING logical input devices:

<0 prompting and echoing is STRING device dependent.

1 display the current STRING value within the echo area.

≥2 reserved for registration.

For all prompt and echo types, the first entry of the string data record is the input buffer size, which is an integer in the range (1..n). This is compared against an implementation defined 'maximum input buffer size' for this device (contained in the workstation description table). If the requested buffer size is greater, the 'maximum input buffer size' is substituted in the stored record. If the initial string is longer than the buffer size, an error is issued.

For all prompt and echo types, the second entry of the string data record is an initial cursor position, which may range from 1 to the length of the initial string plus 1.

When a STRING measure process comes into existence, it obtains a buffer of the current input buffer size. The initial string is copied into the buffer, and the cursor is placed at the initial cursor position within it. Replacement of characters begins at this cursor position.

The items in data records mandatory for each class are: in a STROKE data record, input buffer size in number of points; in a VALUATOR data record, low value and high value; in a STRING data record, input buffer size and initial cursor position. Prompt and echo types which have mandatory values are types 2, 3, 4 and 5 for CHOICE.

8.12.3 Definition of logical input devices

The application program can construct logical input devices of compound types using the function DEFINE LOGICAL INPUT DEVICE. Each workstation supporting input devices provides a number of measures and triggers that map onto physical devices associated with the workstation. The application can specify which of these measures and triggers are to be used in constructing the measure and trigger processes for a specified device. The value returned by such a device is a sequence of values of basic types. Satisfaction of any of the trigger conditions specified causes the corresponding logical input device's trigger to fire.

8.13 Sending messages to a workstation

The MESSAGE function allows a character string to be sent to a workstation. The application program has no control over the position and appearance of the character string and an implementation is allowed to place the string on a device distinct from, but associated with, the workstation.

9 Workstation functions

9.1 Control functions

OPEN WORKSTATION 8.3

In	workstation identifier	WsId
In	workstation type	WsType

GKS requests the operating system to establish a connection for a workstation characterized in the workstation description table by the 'workstation type'. The workstation state list is allocated and initialized as indicated in 10.3.3. The workstation identifier is added to the set of open workstations in the GKS state list.

The current NDC picture is displayed on the workstation.

CLOSE WORKSTATION 6.7.5, 8.3

In	workstation identifier	WsId

The workstation state list is deallocated. The workstation identifier is deleted from the set of open workstations in the GKS state list. The input queue is flushed of all events from all devices on the workstation being closed. If the 'identification of one of the logical input devices that caused an input queue overflow' entry in the GKS error state list refers to this workstation identifier, then all the contents of that entry become undefined.

The connection to the workstation is released. The display surface need not be cleared when CLOSE WORKSTATION is invoked, but it may be cleared.

SET COLOUR MODEL 8.7.6

In	workstation identifier	WsId
In	colour model	ColMod

The 'current colour model' entry in the workstation state list of the specified workstation is set to the value specified by the parameters.

SET REPRESENTATION 8.8

In	workstation identifier	WsId
In	representation designation	Representation
In	index	Rep
In	representation value	VRep

The specified index is associated with the specified representation in the appropriate table on the specified workstation.

SET WORKSTATION WINDOW AND VIEWPORT 8.4

In	workstation identifier	WsId
In	workstation window limits	NDCWindow
In	workstation viewport limits	DCViewport

The 'workstation window' and 'workstation viewport' entries in the workstation state list of the specified workstation are set to the values specified by the parameters.

DEFINE LOGICAL INPUT DEVICE 8.12.3

In	device identifier	DeviceId
In	measure sequence	seq_1 MeasureId
In	trigger set	P Trigger

The measure sequence and trigger set of the specified logical input device are set to the values specified by the parameters. The measure sequence specifies the measures from which the measure process of the logical input device is composed. The logical input values returned by the logical input device consist of a sequence of measure values in the same order.

INITIALIZE LOGICAL INPUT DEVICE 8.12.2

In	device identifier	DeviceId
In	initial values	InitialInput

The initial values are stored in the workstation state list entry for the specified logical input device.

SET WORKSTATION SELECTION CRITERION 8.9

In	workstation identifier	WsId
In	selection type	SelectType
In	selection criterion	SelectCrit

The specified selection criterion on the specified workstation is set to the value specified by the parameter.

COPY PHYSICAL PICTURE TO PHYSICAL METAFILE 8.1

In	workstation identifier	WsId
In	metafile identifier	RenderFileId
In	picture identifier	PictureId

Primitives in the physical picture of the specified workstation are stored in the specified metafile.

COPY BLANK PHYSICAL PICTURE TO PHYSICAL METAFILE		8.1
In	workstation identifier	WsId
In	metafile identifier	RenderFileId
In	picture identifier	PictureId

A blank physical picture is stored in the specified metafile.

COPY PHYSICAL METAFILE PICTURE TO PHYSICAL PICTURE		8.1
In	workstation identifier	WsId
In	metafile identifier	MetafileId
In	picture identifier	PictureId

The specified picture in the specified metafile is added to the physical picture of the specified workstation.

MESSAGE		8.13
In	workstation identifier	WsId
In	message	CharString

The message function:

a) may display a message at an implementation dependent location on the workstation viewport or on some separate device associated with the workstation.

b) does not alter the GKS state list.

c) may affect the workstation in a purely local way (for example, requesting the operator to change paper). Possible effects on the execution of the application program or on subsequent commands sent to the workstation by GKS are stated explicitly in the implementation dependencies manual.

9.2 Inquiry functions

The inquiry functions that retrieve values from the workstation state lists have an input parameter of type RType that can take the following values:

a) SET:	the values returned are those provided by the application program.
b) REALIZED:	the values returned are those used by the workstation when the actual values are mapped to the available values in the workstation.

Inquiries for predefined representations in the workstation description table (see 10) have no such parameter unlike the corresponding inquiries for the representations in the workstation state list (see 10). The values of predefined representations are available on the workstation. Thus all values returned from a predefined representation are such that, if used by an application program to set a representation, a subsequent inquiry for that representation in the workstation state list would return the same values whether SET or REALIZED was specified.

INQUIRE WORKSTATION STATE LIST 6.10

In	workstation identifier	WsId
In	type of returned values	RType
Out	error indicator	ErrorIndicator
Out	workstation state list	WSL

If the inquired information is available, the error indicator is returned as 0 and values are returned in the output parameters.

If the inquired information is not available, the values returned in the output parameters are implementation dependent and the error indicator is set to one of the following error numbers to indicate the reason for non-availability:

2 *GKS not open*
20 *Specified workstation is not open*

INQUIRE WORKSTATION DESCRIPTION TABLE 6.10

In	workstation type	WsType
Out	error indicator	ErrorIndicator
Out	workstation description table	WDT

If the inquired information is available, the error indicator is returned as 0 and values are returned in the output parameters.

If the inquired information is not available, the values returned in the output parameters are implementation dependent and the error indicator is set to one of the following error numbers to indicate the reason for non-availability:

2 *GKS not open*
19 *Specified workstation type not supported*

INQUIRE TEXT EXTENT 8.7.4

In	workstation identifier	WsId
In	text position	WCPoint
In	character string	CharString
Out	error indicator	ErrorIndicator
Out	concatenation point	WCPoint
Out	text extent parallelogram	WCPoint×WCPoint×WCPoint×WCPoint

The concatenation point and text extent parallelogram for the specified TEXT primitive are returned. If the inquired information is available, the error indicator is returned as 0 and values are returned in the output parameters.

If the inquired information is not available, the values returned in the output parameters are implementation dependent and the error indicator is set to one of the following error numbers to indicate the reason for non-availability:

2 *GKS not open*
20 *Specified workstation is not open*

10 GKS data structures

10.1 Notation and data types

In this clause, the contents of the GKS data structures are listed.

The information for each entry includes

a) the name of the entry;

b) the data type;

The notation used to define the data types is described in 7.1.

10.2 Contents of state lists and description tables

10.2.1 Operating state

GKS operating state	GKSOpSt
picture part operating state	PPOpSt
shape operating state	SHOpSt
closed path operating state	CPOpSt

10.2.2 GKS description table

level of GKS	**N**
set of available workstation types	**P** WSType

10.2.3 GKS state list

set of open workstations	**P** WkId
current primitive attributes	PrimAttr ⇸ VPrimAttr
current ASFs	LogAttr ⇸ VAsf
current normalization transformation number	NormTran
normalization transformations	NormTran ⇸ (WCWindow × NDCViewport)
viewport input priorities	seq NormTran
clipping rectangle set	RectangleSet
shielding rectangle set	RectangleSet
clipping indicator	Clip
shielding indicator	Shield
name of the open picture part	PartName
name of the open shape	ShapeName
current shape attributes	ShapeAttr ⇸ VShapeAttr
current sub-path attributes	SubPathAttr ⇸ VSubPathAttr
input queue	seq (DeviceId × InputValue)

10.2.4 Workstation state list

polyline bundle table	PolylineBT
polymarker bundle table	PolymarkerBT
text bundle table	TextBT
fill area bundle table	FillBT
pattern table	PatternBT
current colour model	ColMod

colour table	ColourT
current workstation window	NDCWindow
current workstation viewport	DCViewport
logical input device operating modes	DeviceId → Mode
logical input device initial values	DeviceId → InitialInput
selection criteria	SelectT

10.2.5 Workstation description table

workstation type	WsType
device coordinate units	METRES|OTHER
display space size	
in device units (DC)	**R** × **R**
in raster units	**N** × **N**
vector or raster display	VECTOR|RASTER|OTHER
set of available linetypes	**P** LineType
number of available linewidths	**N**
nominal linewidth (DC)	**R**
minimum linewidth (DC)	**R**
maximum linewidth (DC)	**R**
predefined polyline bundles	PolylineBT
set of available marker types	**P** MarkerType
number of available marker sizes	**N**
nominal marker size (DC)	**R**
minimum marker size (DC)	**R**
maximum marker size (DC)	**R**
predefined polymarker bundles	PolymarkerBT
set of available font and precision pairs	**P** FontPrec
number of available character expansion factors	Expan
minimum character expansion factor	Expan
maximum character expansion factor	Expan
number of available character heights	**N**
minimum character height (DC)	**R**
maximum character height (DC)	**R**
predefined text bundles	TextBT
set of available fill area interior styles	**P** FillInterior
set of available hatch styles	**P** FillStyleInd
predefined fill area bundles	FillBT
predefined pattern representations	PatternBT
set of available colour models	**P** ColMod
default colour model	ColMod
luminance values and chromaticity coefficients of display	LC × LCC × LCC
number of available colours or intensities	**N**
colour available	COLOUR|MONOCHROME
predefined colour representations	ColourT
logical input device initial values	DeviceId → InitialInput

10.2.6 Error state list

error state	OFF|ON
error file	EFile
input device causing queue overflow	DeviceId

10.3 Initial values of state list and description table entries

10.3.1 Operating state

GKS operating state	GKCL
picture part store operating state	PPCL
shape operating state	SHCL
closed path operating state	CPCL

10.3.2 GKS description table

All initial values are implementation dependent.

10.3.3 GKS state list

set of open workstations	empty
current primitive attributes	
POLYLINE INDEX	1
LINETYPE	1
LINEWIDTH SCALE FACTOR	1.0
POLYLINE COLOUR INDEX	1
POLYMARKER INDEX	1
MARKERTYPE	3
MARKER SIZE SCALE FACTOR	1.0
POLYMARKER COLOUR INDEX	1
TEXT INDEX	1
TEXT FONT AND PRECISION	(1, STRING)
CHARACTER EXPANSION FACTOR	1.0
CHARACTER SPACING	1.0
TEXT COLOUR INDEX	1
CHARACTER HEIGHT	0.01
CHARACTER UP VECTOR	(0, 1)
TEXT PATH	RIGHT
TEXT ALIGNMENT	(HNORMAL, VNORMAL)
FILL AREA INDEX	1
FILL AREA INTERIOR STYLE	HOLLOW
FILL AREA STYLE INDEX	1
FILL AREA COLOUR INDEX	1
PATTERN SIZE	(1, 1)
PATTERN REFERENCE POINT	(0, 0)
PICK IDENTIFIER	language binding dependent
NAMESET	empty
current ASFs	all BUNDLED or all INDIVIDUAL
current normalization transformation number	0
normalization transformations	
for each	window (0,1,0,1) viewport (0,1,0,1)
viewport input priorities	(0..n)
0 highest priority	
n lowest (maximum normalization transformation number)	
clipping rectangle set	{(0,1,0,1)}
shielding rectangle set	empty
clipping indicator	CLIP
shielding indicator	NOSHIELD
name of the open picture part	undefined

name of the open shape	undefined
current shape attributes	?
current sub-path attributes	?
input queue	empty

10.3.4 Workstation state list

Initial values are taken from the workstation description table except for the following entries.

current workstation window	(0,1,0,1)
current workstation viewport	(0,xd,0,yd)
where (xd,yd) is the display space size from the workstation description table	
selection criterion	none
logical input device operating modes	REQUEST

10.3.5 Workstation description table

All initial values are implementation dependent.

10.3.6 Error state list

error state	OFF
error file	implementation dependent
input device causing queue overflow	undefined

Annex A

Function lists

(This annex forms an integral part of the standard.)

B. Order of appearance

Control functions (5.2)

OPEN GKS
CLOSE GKS
ESCAPE
EMERGENCY CLOSE GKS
ERROR HANDLING
ERROR LOGGING

Output functions (5.3)

CREATE BUNDLED PRIMITIVE
CREATE GEOMETRIC PRIMITIVE
BEGIN SHAPE
END SHAPE
BEGIN CLOSED PATH
END CLOSED PATH
CREATE SUB PATH
CREATE CONTOUR
SET SUB PATH ATTRIBUTE
SET SHAPE ATTRIBUTE
CREATE CONCATENATED SHAPE

Output attributes (5.4)

SET PRIMITIVE ATTRIBUTE
SET ATTRIBUTE SOURCE FLAG

Normalization transformation functions (5.5)

SET WINDOW AND VIEWPORT
SET CLIPPING RECTANGLE SET
SET SHIELDING RECTANGLE SET
SET VIEWPORT INPUT PRIORITY
SELECT NORMALIZATION TRANSFORMATION
SET CLIPPING INDICATOR
SET SHIELDING INDICATOR

NDC picture functions (5.6)

DELETE PRIMITIVES
REMOVE NAME FROM NDC PICTURE
ADD NAME TO NDC PICTURE

Metafile functions (5.7)

COPY NDC PICTURE TO NDC METAFILE
COPY NDC METAFILE PICTURE TO NDC PICTURE

Picture part store functions (5.8)

BEGIN PICTURE PART
END PICTURE PART
ARCHIVE PICTURE PART
RETRIEVE PICTURE PART FROM ARCHIVE
BEGIN PICTURE PART AGAIN
APPEND PICTURE PART
RENAME PICTURE PART
DELETE PICTURE PART
COPY PICTURE PART TO NDC PICTURE
CREATE IMAGE PICTURE PART

Input functions (5.9)

SET LOGICAL INPUT DEVICE MODE
REQUEST INPUT
SAMPLE INPUT
AWAIT INPUT
FLUSH DEVICE EVENTS

Inquiry functions (5.10)

INQUIRE OPERATING STATE VALUE
INQUIRE GKS DESCRIPTION TABLE
INQUIRE GKS STATE LIST
INQUIRE INPUT QUEUE OVERFLOW

Workstation functions (7)

OPEN WORKSTATION
CLOSE WORKSTATION
SET REPRESENTATION
SET WORKSTATION WINDOW AND VIEWPORT
DEFINE LOGICAL INPUT DEVICE
INITIALIZE LOGICAL INPUT DEVICE
SET WORKSTATION SELECTION CRITERION
COPY PHYSICAL PICTURE TO PHYSICAL METAFILE
COPY BLANK PHYSICAL PICTURE TO PHYSICAL METAFILE
COPY PHYSICAL METAFILE PICTURE TO PHYSICAL PICTURE
MESSAGE
INQUIRE WORKSTATION STATE LIST
INQUIRE WORKSTATION DESCRIPTION TABLE
INQUIRE TEXT EXTENT

Annex B

Error list

(This annex forms an integral part of the standard.)

OPEN GKS

1 GKS is already open

CLOSE GKS

2 GKS not open

ESCAPE

2 GKS not open
3 Specified escape function is not supported

EMERGENCY CLOSE GKS

none

ERROR HANDLING

none

ERROR LOGGING

none

CREATE BUNDLED PRIMITIVE

2 GKS not open

CREATE GEOMETRIC PRIMITIVE

2 GKS not open
33 Specified shape does not exist

BEGIN SHAPE

2 GKS not open
31 Shape already open

END SHAPE

2 GKS not open
32 Shape not open

DELETE SHAPE

2 GKS not open
33 Specified shape does not exist

BEGIN CLOSED PATH

2 GKS not open
29 Closed path already open

END CLOSED PATH
2 GKS not open
30 Closed path not open

CREATE SUB PATH
2 GKS not open
30 Closed path not open

CREATE CONTOUR
2 GKS not open

SET SUB PATH ATTRIBUTE
2 GKS not open

SET SHAPE ATTRIBUTE
2 GKS not open

CREATE CONCATENATED SHAPE
2 GKS not open
33 Specified shape does not exist
34 Invalid concatenation criterion

SET PRIMITIVE ATTRIBUTE
2 GKS not open

SET ATTRIBUTE SOURCE FLAG
2 GKS not open

SET WINDOW AND VIEWPORT
2 GKS not open

SET CLIPPING RECTANGLE SET
2 GKS not open

SET SHIELDING RECTANGLE SET
2 GKS not open

SET VIEWPORT INPUT PRIORITY
2 GKS not open

SELECT NORMALIZATION TRANSFORMATION
2 GKS not open

SET CLIPPING INDICATOR
2 GKS not open

SET SHIELDING INDICATOR
2 GKS not open

DELETE PRIMITIVES
2 GKS not open

REMOVE NAME FROM NDC PICTURE

2 GKS not open

ADD NAME TO NDC PICTURE

2 GKS not open

COPY NDC PICTURE TO NDC METAFILE

2 GKS not open

COPY NDC METAFILE PICTURE TO NDC PICTURE

2 GKS not open
5 Specified picture does not exist in specified metafile

BEGIN PICTURE PART

2 GKS not open
6 Picture part already open
7 Specified picture part name already in use

END PICTURE PART

2 GKS not open
8 Picture part not open

ARCHIVE PICTURE PART

2 GKS not open
9 Specified picture part does not exist

RETRIEVE PICTURE PART FROM ARCHIVE

2 GKS not open
7 Specified picture part name already in use

BEGIN PICTURE PART AGAIN

2 GKS not open
6 Picture part already open
9 Specified picture part does not exist

APPEND PICTURE PART

2 GKS not open
10 Source picture part does not exist
11 Sink picture part does not exist
6 Picture part already open

RENAME PICTURE PART

2 GKS not open
12 Old picture part does not exist
13 New picture part name already in use
14 Old picture part name and new picture part name are the same

DELETE PICTURE PART

2 GKS not open
9 Specified picture part does not exist

COPY PICTURE PART TO NDC PICTURE
- 2 *GKS not open*
- 9 *Specified picture part does not exist*

CREATE IMAGE PICTURE PART
- 2 *GKS not open*
- 7 *Specified picture part name already in use*

SET LOGICAL INPUT DEVICE MODE
- 2 *GKS not open*
- 16 *Specified logical input device does not exist*

REQUEST INPUT
- 2 *GKS not open*
- 16 *Specified logical input device does not exist*

SAMPLE INPUT
- 2 *GKS not open*
- 16 *Specified logical input device does not exist*

AWAIT INPUT
- 2 *GKS not open*
- 17 *Input queue has overflowed*

FLUSH DEVICE EVENTS
- 2 *GKS not open*
- 16 *Specified logical input device does not exist*
- 17 *Input queue has overflowed*

INQUIRE OPERATING STATE VALUE
- *none*

INQUIRE GKS DESCRIPTION TABLE
- *none*

INQUIRE GKS STATE LIST
- *none*

INQUIRE INPUT QUEUE OVERFLOW
- *none*

OPEN WORKSTATION
- 2 *GKS not open*
- 18 *Workstation already open*
- 19 *Specified workstation type not supported*

CLOSE WORKSTATION
- 2 *GKS not open*
- 17 *Input queue has overflowed*
- 20 *Specified workstation is not open*

SET REPRESENTATION
2 *GKS not open*
20 *Specified workstation is not open*
21 *Linetype not supported*
22 *Marker type not supported*
23 *Text font not supported for specified precision*
24 *Specified fill area interior style not supported*
25 *Specified hatch style not supported*

SET WORKSTATION WINDOW AND VIEWPORT
2 *GKS not open*
20 *Specified workstation is not open*

DEFINE LOGICAL INPUT DEVICE
2 *GKS not open*
26 *Specified combination of measures and triggers cannot be provided*
27 *Specified logical input device is active*

INITIALIZE LOGICAL INPUT DEVICE
2 *GKS not open*
16 *Specified logical input device does not exist*
28 *Specified initial value is invalid*

SET WORKSTATION SELECTION CRITERION
2 *GKS not open*
20 *Specified workstation is not open*

COPY PHYSICAL PICTURE TO PHYSICAL METAFILE
2 *GKS not open*
20 *Specified workstation is not open*

COPY BLANK PHYSICAL PICTURE TO PHYSICAL METAFILE
2 *GKS not open*
20 *Specified workstation is not open*

COPY PHYSICAL METAFILE PICTURE TO PHYSICAL PICTURE
2 *GKS not open*
20 *Specified workstation is not open*

MESSAGE
2 *GKS not open*
20 *Specified workstation is not open*

INQUIRE WORKSTATION STATE LIST
none

INQUIRE WORKSTATION DESCRIPTION TABLE
none

INQUIRE TEXT EXTENT
none

System errors

4 *At least one open workstation is unable to generated the specified GDP*
35 *Input queue has not overflowed ever or since the last invocation of INQUIRE INPUT QUEUE OVERFLOW*
36 *Input queue has overflowed, but associated workstation has been closed*

Annex C

Language binding considerations

(This annex does not form an integral part of the standard, but provides additional information.)

Function	Question
OPEN GKS	Can the error file be invalid
CLOSE GKS	—
ESCAPE	Can escape function identification be invalid
	Can escape input data record be invalid
EMERGENCY CLOSE GKS	—
ERROR HANDLING	—
ERROR LOGGING	—
CREATE BUNDLED PRIMITIVE	Is primitive type valid
	Should language binding give errors on:
	Number of points invalid
	Invalid code in string
	Dimensions of colour array invalid
	GDP identifier is invalid
	GDP data record is invalid
CREATE GEOMETRIC PRIMITIVE	Is shape name valid
	Is part name valid
BEGIN SHAPE	Is shape name valid
END SHAPE	—
DELETE SHAPE	Is shape name valid
BEGIN CLOSED PATH	—
END CLOSED PATH	—
CREATE SUB PATH	Are sub-path parameters invalid
CREATE CONTOUR	Is shape name valid
	Are sub-path parameters valid
SET SUB PATH ATTRIBUTE	Is attribute name valid
	Is attribute value valid
SET SHAPE ATTRIBUTE	Is attribute name valid
	Is attribute value valid
CREATE CONCATENATED SHAPE	Is shape name valid
	Is sequence of shape names valid
SET PRIMITIVE ATTRIBUTE	Is attribute name valid
	Is attribute value valid
SET ATTRIBUTE SOURCE FLAG	Is attribute name valid
SET WINDOW AND VIEWPORT	Transformation number not positive
	Transformation number greater than 31
	Rectangle definition is invalid
SET CLIPPING RECTANGLE SET	Clipping rectangle definition is invalid
	Invalid set
SET SHIELDING RECTANGLE SET	Shielding rectangle definition is invalid
	Invalid set

SET VIEWPORT INPUT PRIORITY	Transformation number negative
	Transformation number greater than 31
SELECT NORMALIZATION TRANSFORMATION	Transformation number negative
	Transformation number greater than 31
SET CLIPPING INDICATOR	—
SET SHIELDING INDICATOR	—
DELETE PRIMITIVES	—
REMOVE NAME FROM NDC PICTURE	Invalid name
ADD NAME TO NDC PICTURE	Invalid name
COPY NDC PICTURE TO NDC METAFILE	Invalid name
COPY NDC METAFILE PICTURE TO NDC PICTURE	Invalid name
BEGIN PICTURE PART	Invalid name
END PICTURE PART	—
ARCHIVE PICTURE PART	Invalid name
RETRIEVE PICTURE PART FROM ARCHIVE	Invalid name
BEGIN PICTURE PART AGAIN	Invalid name
APPEND PICTURE PART	Invalid names
RENAME PICTURE PART	Invalid names
DELETE PICTURE PART	Invalid name
COPY PICTURE PART TO NDC PICTURE	Invalid name(s)
CREATE IMAGE PICTURE PART	Invalid name
	Invalid image function
SET LOGICAL INPUT DEVICE MODE	Invalid device identifier
REQUEST INPUT	Invalid device identifier
SAMPLE INPUT	Invalid device identifier
AWAIT INPUT	Invalid timeout
FLUSH DEVICE EVENTS	Invalid device identifier
INQUIRE OPERATING STATE VALUE	—
INQUIRE GKS DESCRIPTION TABLE	—
INQUIRE GKS STATE LIST	—
INQUIRE INPUT QUEUE OVERFLOW	—
OPEN WORKSTATION	Invalid identifier
	Invalid workstation type
	Connection identifier required?
CLOSE WORKSTATION	Invalid identifier
SET REPRESENTATION	Invalid identifier
	Invalid representation
	Invalid index
	Linetype is equal to 0
	Linewidth scale factor is less than zero
	Marker type is equal to 0
	Marker size scale factor is less than 0
	Text font is equal to 0
	Character expansion factor is less than or equal to 0
	Colour index is invalid
	Pattern index is invalid
	Dimensions of colour array are invalid
	Colour is outside range [0,1]
SET WORKSTATION WINDOW AND VIEWPORT	Invalid workstation identifier
	Invalid window limits

	Invalid viewport limits
DEFINE LOGICAL INPUT DEVICE	Invalid device identifier
	Invalid measure sequence
	Invalid trigger set
INITIALIZE LOGICAL INPUT DEVICE	Invalid device identifier
	Rectangle definition is invalid
SET WORKSTATION SELECTION CRITERION	Invalid workstation identifier
COPY PHYSICAL PICTURE	Invalid workstation identifier
TO PHYSICAL METAFILE	Invalid metafile identifier
	Invalid picture identifier
COPY BLANK PHYSICAL PICTURE	Invalid workstation identifier
TO PHYSICAL METAFILE	Invalid metafile identifier
	Invalid picture identifier
COPY PHYSICAL METAFILE PICTURE	Invalid workstation identifier
TO PHYSICAL PICTURE	Invalid metafile identifier
	Invalid picture identifier
MESSAGE	Invalid workstation identifier
INQUIRE WORKSTATION STATE LIST	Invalid workstation identifier
INQUIRE WORKSTATION DESCRIPTION TABLE	Invalid workstation type
INQUIRE TEXT EXTENT	Invalid workstation identifier
	Invalid code in string

Where the specific escape function identification is bound to an integer in a programming language, specific escape function identifications greater than 0 are reserved for registration and specific escape function identifications less than 0 are implementation dependent.

Where the GDP identifier is bound to an integer in a programming language, GDP identifiers greater than 0 are reserved for registration and GKD identifiers less than 0 are implementation dependent.

Annex D

Allowable differences

(This annex does not form an integral part of the standard, but provides additional information.)

Annex E

Relationship to CGM

(This annex does not form an integral part of the standard, but provides additional information.)

Annex F

Relationship to CGI

(This annex does not form an integral part of the standard, but provides additional information.)

Annex G

Function summary

(This annex does not form an integral part of the standard, but provides additional information.)

Annex H

CIE colour model

(This annex does not form an integral part of the standard, but provides additional information.)
As ISO 9592-1: 1989 (Programmer's Hierarchical Interactive Graphics System) Annex I.