

IC 97

International Standard



7942

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION

Information processing systems — Computer graphics — Graphical Kernel System (GKS) functional description

Systèmes de traitement de l'information — Langage de programmation graphique — GKS (Graphical Kernel System) description fonctionnelle

First edition — 1985-08-15

UDC 681.3.06 : 003.6

Ref. No. ISO 7942-1985 (E)

Descriptors : data processing, information interchange, graphic data processing, programming (computers), programming languages, computer graphics.



Documents

Document Sharing Hub

Price based on 245 pages

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Every member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work.

Draft International Standards adopted by the technical committees are circulated to the member bodies for approval before their acceptance as International Standards by the ISO Council. They are approved in accordance with ISO procedures requiring at least 75% approval by the member bodies voting.

International Standard ISO 7942 was prepared by Technical Committee ISO/TC 97, *Information processing systems*.

Contents

	Page
0 Introduction	1
1 Scope and field of application.....	3
2 References	4
3 Definitions.....	5
4 The Graphical Kernel System.....	9
4.1 About this standard	9
4.1.1 Specification.....	9
4.1.2 Registration	9
4.2 Introduction to GKS.....	10
4.3 Concepts.....	12
4.4 Graphical output.....	14
4.4.1 Output primitives.....	14
4.4.2 Output primitive attributes	14
4.4.3 Polyline attributes	18
4.4.4 Polymarker attributes	19
4.4.5 Text attributes.....	19
4.4.6 Fill area attributes	28
4.4.7 Cell array attributes	29
4.4.8 Generalized Drawing Primitive attributes.....	29
4.4.9 Colour.....	29
4.5 Workstations	31
4.5.1 Workstation characteristics.....	31
4.5.2 Selecting a workstation.....	32
4.5.3 Deferring picture changes	33
4.5.4 Clearing the display surface.....	35
4.5.5 Elimination of primitives outside segments.....	36
4.5.6 Sending messages to a workstation.....	36
4.6 Coordinate systems and transformations	37
4.6.1 Normalization transformations.....	37
4.6.2 Clipping	37
4.6.3 Workstation transformations.....	38
4.6.4 Transformation of locator input.....	40
4.6.5 Transformation of stroke input	42
4.7 Segments.....	43
4.7.1 Introduction to segments	43
4.7.2 Segment attributes	44

4.7.3	Segment transformations	44
4.7.4	Clipping and WDSS	45
4.7.5	Workstation Independent Segment Storage	45
4.7.6	WISS functions and clipping	45
4.8	Graphical input	47
4.8.1	Introduction to logical input devices	47
4.8.2	Logical input device model	48
4.8.3	Operating modes of logical input devices	49
4.8.4	Measures of each input class	51
4.8.5	Input queue and current event report	52
4.8.6	Initialisation of input devices	52
4.9	GKS Metafile interface	54
4.10	GKS levels.....	56
4.10.1	Introduction to levels	56
4.10.2	The level structure	56
4.10.3	Level functionality	57
4.11	States of GKS and inquiry functions	61
4.11.1	Description of states	61
4.11.2	Inquiry functions	62
4.12	Error handling	63
4.13	Special interfaces between GKS and the application program ..	65
5	GKS functions	66
5.1	Notational conventions	66
5.2	Control functions	67
5.3	Output functions	74
5.4	Output attributes	80
5.4.1	Workstation independent primitive attributes	80
5.4.2	Workstation attributes (representations)	90
5.5	Transformation functions	96
5.5.1	Normalization transformation	96
5.5.2	Workstation transformation	97
5.6	Segment functions	99
5.6.1	Segment manipulation functions	99
5.6.2	Segment attributes	103
5.7	Input functions	106
5.7.1	Initialisation of input devices	106
5.7.2	Setting the mode of input devices	113
5.7.3	Request input functions	116
5.7.4	Sample input functions	119
5.7.5	Event input functions	122
5.8	Metafile functions	126
5.9	Inquiry functions	128
5.9.1	Introduction to inquiry functions	128
5.9.2	Inquiry function for operating state value	128
5.9.3	Inquiry functions for GKS description table	128
5.9.4	Inquiry functions for GKS state list	130
5.9.5	Inquiry functions for workstation state list	135
5.9.6	Inquiry functions for workstation description table	152
5.9.7	Inquiry functions for segment state list	169
5.9.8	Pixel inquiries	170
5.9.9	Inquiry function for GKS error state list	172
5.10	Utility functions	173
5.11	Error handling	174
6	GKS data structures	176
6.1	Notation and data types	176
6.2	Operating state	178
6.3	GKS description table	179

6.4	GKS state list	180
6.5	Workstation state list	182
6.6	Workstation description table	185
6.7	Segment state list	189
6.8	GKS error state list	190

Annexes

A	Function lists	191
A.1	Alphabetic	191
A.2	Order of appearance	194
A.3	Ordered by level	199
A.3.1	Level 0a	199
A.3.2	Level 0b	200
A.3.3	Level 0c	201
A.3.4	Level 1a	201
A.3.5	Level 1b	202
A.3.6	Level 1c	202
A.3.7	Level 2a	202
A.4	Ordered by state	202
A.4.1	Functions allowed in state GKCL	202
A.4.2	Functions allowed in state GKOP	203
A.4.3	Functions not allowed in state WSOP	203
A.4.4	Functions not allowed in state WSAC	204
A.4.5	Functions not allowed in state SGOP	204
A.5	Applicability to workstation groups	204
B	Error list	208
B.1	Implementation dependent	208
B.2	States	208
B.3	Workstations	208
B.4	Transformations	208
B.5	Output attributes	209
B.6	Output primitives	209
B.7	Segments	209
B.8	Input	210
B.9	Metafiles	210
B.10	Escape	210
B.11	Miscellaneous	210
B.12	System	210
B.13	Reserved errors	210
C	Interfaces	212
C.1	General	212
C.2	Language binding	212
C.3	Implementation	213
D	Allowable differences in GKS implementations	214
D.1	General	214
D.2	Global differences	214
D.3	Workstation dependent differences	215
E	Metafile structure	217
E.1	Metafiles	217
E.1.1	General	217
E.1.2	ISO 8632 Metafile	217
E.1.3	Metafile designed for GKS	217
E.2	File format and data format	218
E.3	Generation of metafiles	219
E.4	Interpretation of metafiles	222
E.4.1	General	222

	E.4.2	Control items.....	222
	E.4.3	Output primitives.....	222
	E.4.4	Output primitive attributes.....	222
	E.4.5	Workstation attributes.....	222
	E.4.6	Transformations.....	222
	E.4.7	Segment manipulation.....	222
	E.4.8	Segment attributes.....	222
	E.5	Control items.....	222
	E.6	Items for output primitives.....	225
	E.7	Items for output primitive attributes.....	226
	E.8	Items for workstation attributes.....	228
	E.9	Items for transformations.....	230
	E.10	Items for segment manipulation.....	230
	E.11	Items for segment attributes.....	230
	E.12	User items.....	231
F		Sample programs.....	232
G		GKS functions summary.....	239
	G.1	Control functions.....	239
	G.2	Output functions.....	239
	G.3	Output attributes.....	240
	G.3.1	Workstation independent primitive attributes.....	240
	G.3.2	Workstation attributes (representations).....	241
	G.4	Transformation functions.....	241
	G.4.1	Normalization transformation.....	241
	G.4.2	Workstation transformation.....	241
	G.5	Segment functions.....	242
	G.5.1	Segment manipulation functions.....	242
	G.5.2	Segment attributes.....	242
	G.6	Input functions.....	242
	G.6.1	Initialisation of input devices.....	242
	G.6.2	Setting mode of input devices.....	243
	G.6.3	Request input functions.....	243
	G.6.4	Sample input functions.....	243
	G.6.5	Event input functions.....	244
	G.7	Metafile functions.....	244
	G.8	Inquiry functions.....	244
	G.9	Utility functions.....	245
	G.10	Error handling.....	245

Information processing systems — Computer graphics — Graphical Kernel System (GKS) functional description

0 Introduction

The Graphical Kernel System (GKS) provides a set of functions for computer graphics programming. GKS is a basic graphics system that can be used by the majority of applications that produce computer generated pictures.

The main reasons for standardizing basic computer graphics are:

- a) to allow application programs involving graphics to be easily portable between different installations;
- b) to aid the understanding and use of graphics methods by application programmers;
- c) to serve manufacturers of graphics equipment as a guideline in providing useful combinations of graphics capabilities in a device.

In order to reach these main objectives, the GKS design was based on the following requirements:

- d) GKS should include all the capabilities that are essential for the whole spectrum of graphics, from simple passive output to highly interactive applications.
- e) The whole range of graphics devices, including vector and raster devices, microfilm recorders, storage tube displays, refresh displays and colour displays should be controllable by GKS in a uniform way.
- f) GKS should provide all the capabilities required by a majority of applications without becoming unduly large.

These requirements were used to formulate a number of principles that were used to judge specific design alternatives. Thus it was possible to contribute to the overall design goals while focussing on certain aspects. Five design aspects were identified, each having a group of principles

g) Design goals: The following principles should not be violated by any technical design:

- 1) consistency: the mandatory requirements of GKS should not be mutually contradictory;
- 2) compatibility: other standards or commonly accepted rules of practice should not be violated;
- 3) orthogonality: the functions or modules of GKS should be independent of each other, or the dependency should be structured and well defined.

h) Functional capabilities: The following principles were used to define the extent of GKS:

- 1) completeness: all functions that a majority of applications want to use on a given level of functionality should be included;
- 2) minimality: functions that are unnecessary for applications of a given level of functionality should not be provided;
- 3) compactness: an application should be able to achieve a desired result by a set of functions and parameters that is as small as possible;

4) richness: a rich set of functions offers an extensive range of facilities that stretches beyond the basic functions and includes higher order capabilities.

It is obvious that there is a trade off between the principles in this group. Therefore, the functions of GKS are organized in nine levels. An implementation of GKS provides precisely the functions of one of these levels. While the lowest level contains only a minimal set of functions, higher levels are allowed to extend beyond the basic needs towards greater richness.

i) User interface design: The following principles were used to define the user interface design:

- 1) user friendliness: GKS should allow the design of a desirable user interface;
- 2) clarity: the concepts and functional capabilities of GKS should be easily understandable, especially by the application programmer;
- 3) error handling: failure of system functions or modules, caused by errors of the system itself or by the application program, should be treated in such a way that the error reaction is clearly understandable and informative to the application programmer and that the impact on the system and the application program is as small as possible.

Clarity and sound error handling are essential parts of user friendliness. Error handling is an integral part of GKS. To aid clarity, the system and its state can be presented to the user in an easily comprehensible manner.

Clarity applies not only to the system design but also to the system description. To this end, the GKS specification is divided into a general description, a description of the underlying logical data structures representing the state of the system, and a description of the functions and their effects on these data structures.

j) Graphics devices: The following principles are associated with the range of graphics devices that can be addressed by GKS:

- 1) device independence: GKS functions should be designed to allow an application program, using these functions, to address facilities of quite different graphics output and input devices without modification of the program structure;
- 2) device richness: the full capabilities of a wide range of different graphics output and input devices should be accessible from the functions of GKS.

These principles led to a fundamental concept underlying the GKS architecture: the concept of multiple independent graphical workstations connected to and driven by GKS. The application program can inquire the capabilities of every workstation. The GKS design includes escape functions that are easily identifiable within an application program and can be used to access special facilities of a particular device.

k) Implementation: The last group of principles is related to the implementation of GKS:

- 1) implementability: it should be possible to support the GKS functions in most host languages, on most operating systems and with most graphics devices;
- 2) language independence: it should be possible to access the standard facilities of GKS from all ISO standard programming languages;
- 3) efficiency: GKS should be capable of being implemented without time consuming algorithms;
- 4) robustness: the operator and application programmer should be protected in the best possible way from hardware or software failure of the system.

The five groups of principles are interconnected. For example, design goals and functional capabilities both contribute to user friendliness. Efficiency is also important when considering response time in an interactive environment. Some principles may be conflicting, such as richness versus minimality, comprehensive error handling versus efficiency, and compactness versus device richness. Compromises needed to be made to achieve the overall design objective: GKS should have an easily comprehensible structure and a set of functions that enables a vast majority of computer graphics users to design portable, device independent application programs addressing the whole range of computer graphics equipment.

1 Scope and field of application

This International Standard specifies a set of functions for computer graphics programming, the Graphical Kernel System (GKS). GKS is a basic graphics system for applications that produce computer generated two dimensional pictures on line graphics or raster graphics output devices. It supports operator input and interaction by supplying basic functions for graphical input and picture segmentation. It allows storage and dynamic modification of pictures. A fundamental concept in GKS is the workstation, consisting of a number of input devices and a single output device. Several workstations can be used simultaneously. The application program is allowed to adapt its behaviour at a workstation to make best use of workstation capabilities. This International Standard includes functions for storage on and retrieval from an external graphics file. Last but not least, the functions are organized in upward compatible levels with increasing capabilities.

NOTE - For certain parameters of the functions, GKS defines value ranges as being reserved for registration (see 4.1.2). The meanings of these values will be defined using the established procedures.

GKS defines a language independent nucleus of a graphics system. For integration into a programming language, GKS is embedded in a language dependent layer obeying the particular conventions of that language.

Annexes C to G are given for information; they do not form part of the specification.

2 References

ISO 646, *Information processing - ISO 7-bit coded character set for information interchange.*

ISO 2022, *Information processing - ISO 7-bit and 8-bit coded character sets - Code extension techniques.*

ISO 2382/13, *Data processing - Vocabulary - Part 13: Computer graphics.*

ISO 6093, *Information processing - Representation of numerical values in character strings for information interchange.*¹⁾

ISO 8632, *Information processing systems - Computer Graphics - Metafile for transfer and storage of picture description information*

- *Part 1 : Functional description.*¹⁾
- *Part 2 : Character encoding.*¹⁾
- *Part 3 : Binary encoding.*¹⁾
- *Part 4 : Clear text encoding.*¹⁾

¹⁾ At present at the stage of draft.

3 Definitions

This clause gives the definition of the important terms of the Graphical Kernel System (GKS).

NOTE - As far as possible, commonly accepted graphics terminology is used.

3.1 acknowledgement: Output to the operator of a logical input device indicating that a trigger has fired.

3.2 aspect ratio: A ratio of x to y used to describe the shape of a rectangle in a particular coordinate system (for example, of a workstation window or a workstation viewport).

3.3 aspects of primitives: Ways in which the appearance of a primitive can vary. Some aspects are controlled directly by primitive attributes, some are controlled indirectly through a bundle table. Primitives inside segments have an aspect controlled through the segment containing them, for example highlighting; primitives outside segments do not.

3.4 attribute: A particular property that applies to a display element (output primitive) or a segment. Examples: highlighting, character height. In GKS, some properties of workstations are called workstation attributes.

3.5 baseline: A horizontal line within a character body (see figure 3) 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.

3.6 bundle index: An index into a bundle table for a particular output primitive. It defines the workstation dependent aspects of the primitive.

3.7 bundle table: A workstation dependent table associated with a particular output primitive. Entries in the table specify all the workstation dependent aspects of a primitive. In GKS, bundle tables exist for the following output primitives: polyline, polymarker, text and fill area.

3.8 capline: A horizontal line within a character body (see figure 3) 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 GKS output primitive consisting of a rectangular grid of equal size rectangular cells, each having a single colour.

NOTE - These cells do not necessarily map one-to-one with pixels.

3.10 centreline: A vertical line bisecting the character body (see figure 3).

3.11 character body: A rectangle used by a font designer to define a character shape (see figure 3). All character bodies in a font have the same height.

3.12 choice device: A GKS logical input device providing a non-negative integer defining one of a set of alternatives.

3.13 clipping: Removing parts of display elements that lie outside a given boundary, usually a window or viewport.

3.14 colour table: A workstation dependent table, in which the entries specify the values of the red, green and blue intensities defining a particular colour.

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

3.16 device coordinate (DC): A coordinate expressed in a coordinate system that is device dependent. In GKS, DC units are metres on a device capable of producing a precisely scaled image and appropriate workstation dependent units otherwise.

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

3.18 device space: The space defined by the addressable points of a display device.

3.19 display device; graphics device: A device (for example refresh display, storage tube display, plotter) on which display images can be represented.

- 3.20 display image; picture:** A collection of output primitives or segments that are represented together at any one time on a display surface.
- 3.21 display space:** (1) That portion of the device space corresponding to the area available for displaying images. (2) The working space of an input device such as a digitiser.
- 3.22 display surface; view surface:** In a display device, that medium on which display images may appear.
- 3.23 echo:** The immediate notification of the current value provided by an input device to the operator at the display console.
- 3.24 escape:** A function in GKS used to access implementation or device dependent features, other than for the generation of graphical output, that are not otherwise addressed by GKS.
- 3.25 feedback:** Output indicating to the operator the application program's interpretation of a logical input value.
- 3.26 fill area:** A GKS output primitive consisting of a polygon (closed boundary) which may be hollow or may be filled with a uniform colour, a pattern, or a hatch style.
- 3.27 fill area bundle table:** A table associating specific values for all workstation dependent aspects of a fill area primitive with a fill area bundle index. In GKS, this table contains entries consisting of interior style, style index, and colour index.
- 3.28 Generalized Drawing Primitive (GDP):** An output primitive used to address special geometrical workstation capabilities such as curve drawing.
- 3.29 GKS level:** Two values in the range 0 to 2 and a to c which together define a set of functional capabilities of GKS. An implementation of GKS provides precisely the functions of one level.
- 3.30 GKS Metafile (GKSM):** A sequential file that can be written or read by GKS and is used for long-term storage (and for transmittal and transferral) of graphical information.
- 3.31 halfline:** A horizontal line between the capline and the baseline within the character body (see figure 3), 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.
- 3.32 hatch:** One possible method of filling the interior of a polygon specified by a fill area primitive. The interior is filled with an arrangement of one or more sets of parallel lines.
- 3.33 highlighting:** A device independent way of emphasizing a segment by modifying its visual attributes. For example, blinking.
- 3.34 implementation mandatory:** Implementation mandatory describes a property that is required to be realized identically on all workstations of all implementations of GKS.
- 3.35 input class:** A set of input devices that are logically equivalent with respect to their function. In GKS, the input classes are: LOCATOR, STROKE, VALUATOR, CHOICE, PICK and STRING.
- 3.36 inquiry function:** A GKS function whose purpose is to return values depending on the current state of GKS or on some fixed property of the GKS implementation. There is no effect on the state of GKS or on the display image.
- 3.37 locator device:** A GKS logical input device providing a position in world coordinates and a normalization transformation number.
- 3.38 logical input device:** A logical input device is an abstraction of one or more physical devices that delivers logical input values to the program. Logical input devices in GKS can be of type LOCATOR, STROKE, VALUATOR, CHOICE, PICK and STRING.
- 3.39 logical input value:** A value delivered by a logical input device.
- 3.40 marker:** A glyph with a specified appearance which is used to identify a particular location.
- 3.41 measure:** A value (associated with a logical input device), which is determined by one or more physical input devices and a mapping from the values delivered by the physical devices. The logical input value delivered by a logical input device is the current value of the measure.
- 3.42 MI:** An abbreviation for GKS metafile input, a category of workstation.
- 3.43 MO:** An abbreviation for GKS metafile output, a category of workstation.
- 3.44 normalization transformation; viewing transformation; window-to-viewport transformation:** A transformation that maps the boundary and interior of a window to the boundary and interior of a viewport. In GKS

Definitions

this transformation maps positions in world coordinates to normalized device coordinates.

3.45 normalized device coordinates (NDC): A coordinate specified in a device independent intermediate coordinate system, normalized to some range, typically 0 to 1. In GKS, during an intermediate state the coordinates may lie outside the defined range, but associated clipping information ensures that the output does not exceed the coordinate range $[0,1] \times [0,1]$.

3.46 operator: Person manipulating physical input devices so as to change the measures of logical input devices and cause their triggers to fire.

3.47 output primitive; graphic primitive; display element: A basic graphic element that can be used to construct a display image. Output primitives in GKS are polyline, polymarker, text, fill area, cell array, and generalized drawing primitive.

3.48 pick device: A GKS logical input device providing the pick identifier attached to an output primitive and the associated segment name.

3.49 pick identifier: A name, attached to individual output primitives within a segment, and returned by the pick device. The same pick identifier can be assigned to different output primitives.

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

3.51 polyline: A GKS output primitive consisting of a set of connected lines.

3.52 polyline bundle table: A table associating specific values for all workstation dependent aspects of a polyline primitive with a polyline bundle index. In GKS, this table contains entries consisting of linetype, linewidth scale factor, and colour index.

3.53 polymarker: A GKS output primitive consisting of a set of locations, each to be indicated by a marker.

3.54 polymarker bundle table: A table associating specific values for all workstation dependent aspects of a polymarker primitive with a polymarker bundle index. In GKS, this table contains entries consisting of marker type, marker size scale factor, and colour index.

3.55 primitive attribute: Primitive attribute values (for output primitives) are selected by the application in a workstation independent manner, but can have workstation dependent effects.

3.56 prompt: Output to the operator indicating that a specific logical input device is available.

3.57 raster graphics: Computer graphics in which a display image is composed of an array of pixels arranged in rows and columns.

3.58 rotation: Turning all or part of a display image about an axis. In GKS, this capability is restricted to segments.

3.59 scaling; zooming: Enlarging or reducing all or part of a display image by multiplying the coordinates of display elements by a constant value. In GKS, this capability is restricted to segments.

NOTE - For different scaling in two orthogonal directions two constant values are required.

3.60 segment: A collection of display elements that can be manipulated as a unit.

3.61 segment attributes: Attributes that apply only to segments. In GKS, segment attributes are visibility, highlighting, detectability, segment priority, and segment transformation.

3.62 segment priority: A segment attribute used to determine which of several overlapping segments takes precedence for graphic output and input.

3.63 segment transformation: A transformation that causes the display elements defined by a segment to appear with varying position (translation), size (scale), and/or orientation (rotation) on the display surface.

3.64 string device: A GKS logical input device providing a character string as its result.

3.65 stroke device: A GKS logical input device providing a sequence of points in world coordinates, and a normalization transformation number.

3.66 text: A GKS output primitive consisting of a character string.

3.67 text bundle table: A table associating specific values for all workstation dependent aspects of a text primitive with a text bundle index. In GKS, this table contains entries consisting of text font and precision, character expansion factor, character spacing and colour index.

3.68 text font and precision: An aspect of text in GKS, having two components, font and precision, which together determine the shape of the characters being output, on a particular workstation. In addition, the precision describes the fidelity with which the other text aspects match those requested by an application program. In order of increasing fidelity, the precisions are: STRING, CHAR and STROKE.

3.69 translation; shift: The application of a constant displacement to the position of all or part of a display image. In GKS, this capability is restricted to segments.

3.70 trigger: A physical input device or set of devices that an operator can use to indicate significant moments in time.

3.71 valuator device: A GKS logical input device providing a real number.

3.72 viewport: An application program specified part of normalized device coordinate space. In GKS, this definition is restricted to a rectangular region of normalized device coordinate space used in the definition of the normalization transformation.

3.73 window: A predefined part of a virtual space. In GKS, this definition is restricted to a rectangular region of the world coordinate space used for the definition of the normalization transformation.

3.74 workstation: GKS is based on the concept of abstract graphical workstations, which provide the logical interface through which the application program controls physical devices.

3.75 Workstation Dependent Segment Storage (WDSS): Segment storage on a workstation that is used for graphical output. Segments cannot be transferred from WDSS to another workstation.

3.76 Workstation Independent Segment Storage (WISS): A special workstation type, where segments can be stored and later transferred to other workstations.

3.77 workstation mandatory: Workstation mandatory describes a property that is required to be realized identically on all workstations of a GKS implementation.

3.78 workstation transformation: A transformation that maps the boundary and interior of a workstation window into the boundary and interior of a workstation viewport (part of display space), preserving aspect ratio. In GKS, this transformation maps positions in normalized device coordinates to device coordinates. The effect of preserving aspect ratio is that the interior of the workstation window may not map to the whole of the workstation viewport.

3.79 workstation viewport: A portion of display space currently selected for output of graphics.

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

3.81 world coordinate (WC): A device independent Cartesian coordinate system used by the application program for specifying graphical input and output.

4 The Graphical Kernel System

4.1 About this standard

4.1.1 Specification

The set of functions known as the Graphical Kernel System shall be as described in clauses 4, 5 and 6 and annexes A and B. These functions are organized in nine upward compatible levels with increasing capabilities as described in 4.10. An implementation of GKS shall implement precisely the functions of one level. A GKS implementation shall be invalid if it lies between two defined levels or outside the defined levels. In an implementation, all graphical capabilities that can be addressed by GKS functions shall be used only via GKS.

4.1.2 Registration¹⁾

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.

¹⁾ 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.

4.2 Introduction to GKS

The Graphical Kernel System (GKS) provides a functional interface between an application program and a configuration of graphical input and output devices. The functional interface contains all basic functions for interactive and non-interactive graphics on a wide variety of graphics equipment.

The interface is at such a level of abstraction that hardware peculiarities are shielded from the application program. As a result a simplified interface presenting uniform output primitives (POLYLINE, POLY-MARKER, TEXT, FILL AREA, CELL ARRAY, GENERALIZED DRAWING PRIMITIVE), and uniform input classes (LOCATOR, STROKE, VALUATOR, CHOICE, PICK, STRING) is obtained.

In 4.3 the concepts of basic output, input and the organization of input and output sequences are outlined. A central concept both for structuring GKS and for realizing device independence is introduced, called the workstation.

The facilities for picture manipulation and change are introduced via the segment facilities, the dynamic attributes and the transformations. The integral control over all these methods for change is further explained in 4.5.3 on workstations.

The concept of multiple workstations allows simultaneous output to and input from various display systems. Facilities for internal and external storage are provided by special workstations together with the possibility of transferring graphical entities directly from the special workstation for internal storage to other workstations.

Not every GKS implementation needs to support the full set of functions. Nine levels are defined to meet the different requirements of graphics systems. Each GKS implementation precisely provides the functions of one level. The levels are upward compatible.

GKS defines only a language independent nucleus of a graphics system. For integration into a language, GKS is embedded in a language dependent layer containing the language conventions, for example, parameter and name assignment.

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

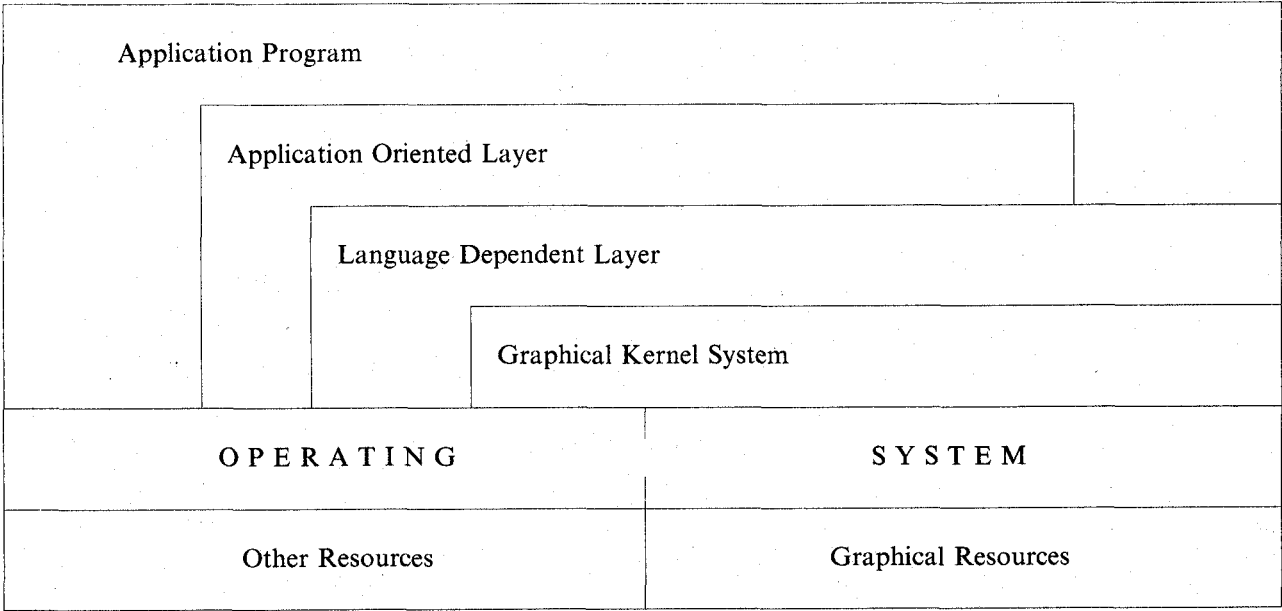


Figure 1 - Layer model of GKS

4.3 Concepts

The graphical output that is generated by GKS is built up from two groups of basic elements called output primitives and primitive attributes. The output primitives are abstractions of basic actions a device can perform, such as drawing lines, and printing character strings. The attributes control the aspects of the output primitives on a device, such as linestyle, colour, character height and pick identifier. Non-geometric aspects, such as colour but not character height, can be controlled for each workstation individually, to make best use of its capabilities.

The graphical information that is input from a device, as a result of operator actions, is mapped by GKS onto six classes of input each represented by a data type referred to as a logical input value. An instance of such a device representation is called a logical input device. The effect of input actions on the display surface, such as prompts and echoes, is controlled by GKS for each logical input device individually.

The two abstract concepts (abstract output and abstract input) are the building blocks of a so-called abstract workstation. A workstation of GKS represents a unit consisting of zero or one display surfaces and zero or more input devices, such as keyboard, tablet and lightpen. The workstation presents these devices to the application program as a configuration of abstract devices thereby shielding the hardware peculiarities.

The geometrical information (coordinates) contained in output primitives, attributes and logical input values (locators and strokes) can be subjected to transformations. These transformations perform mappings between three coordinate systems, namely:

- a) World Coordinates (WC), used by the application programmer;
- b) Normalized Device Coordinates (NDC), used to define a uniform coordinate system for all workstations;
- c) Device Coordinates (DC), one coordinate system per workstation, representing its display space coordinates.

Output primitives and attributes are mapped from WC to NDC by normalization transformations, from NDC to NDC by segment transformations (see next paragraph), and from NDC to DC by workstation transformations. Locator input is mapped by an inverse workstation transformation from DC to NDC and by one of the inverse normalization transformations from NDC to WC.

Output primitives and primitive attributes may be grouped together in a segment. Segments are the units for manipulation and change. Manipulation includes creation, deletion, and renaming. Change includes transforming a segment, making a segment visible or invisible, and highlighting a segment. Segments also form the basis for workstation independent storage of pictures at run time. Via this storage, which is set up as a special workstation called workstation independent segment storage, segments can be inserted and transferred to other workstations.

The attributes which control the appearance of parts of the picture (output primitives, segments, prompt and echo types of input devices) on the display surface are organized in a uniform manner. Two groups of attributes apply to the appearance of each output primitive: primitive attributes (that are workstation independent) and workstation attributes. Primitive attributes are specified modally and are bound to a primitive when it is created. The primitive attributes include all geometric aspects of primitives, such as character height for text and pattern size for fill area. In addition, the non-geometric aspects of primitives are controlled by the primitive attributes in one of two ways. Either a single attribute is used to specify all the non-geometric aspects of the primitive by an index which points to a workstation dependent representation (set of values) or one attribute is used to specify each of the non-geometric aspects of the primitive in a workstation independent way. The former is referred to as bundled specification and the latter is referred to as individual specification.

Workstation attributes include the actual representations on a workstation pointed to by indices used in bundled specification of non-geometric aspects. For example, the representations (or bundles) for polyline each contain values of linestyle, linewidth scale factor and colour index. Workstation attributes also specify the colour and pattern tables and the control over deferral of picture change. Workstation attributes can be reset dynamically.

The Graphical Kernel System

Concepts

The appearance of segments is controlled by segment attributes, which are segment transformation, visibility, highlighting, segment priority, and detectability. These may be reset dynamically. Segment attributes can be a basis for feedback during manipulations (for example, highlighting).

The attributes which control the operation of logical input devices can be specified either upon initialisation or as part of input device setting, depending upon the attributes. Through initialisation, an initial value, a prompt and echo technique, and an area on the screen for echoing can be specified. A data record may further provide device specific attributes. Through input device setting, the operating mode may be selected and the echo may be switched on or off. The operating modes of logical input devices specify who (operator or application program) has the initiative: SAMPLE input is acquired directly by the application program; REQUEST input is produced by the operator in direct response to the application program; EVENT input is generated asynchronously by the operator and is collected in a queue for the application program.

At run time GKS can be in one of five different operating states. Associated with each state are the set of GKS functions allowed in this state, and a set of state variables. The operating state concept and the state variables allow for proper specification of initialisations (for example, at OPEN WORKSTATION) and the effect of various functions, especially with respect to the maintenance of device independence. One special set of functions called inquiry functions is allowed in all states. They give read-only access to the state lists. In this way useful information can be provided when errors occur. Other inquiry functions allow read-only access to the workstation descriptions, to allow the application program to adapt to particular workstation capabilities. Inquiry functions never cause errors. Instead they return information specifying whether a valid inquiry was made.

GKS provides an interface to a system for filing graphical information for the purpose of external long term storage and exchange. The interface consists of a GKS Metafile output workstation, which writes to a so-called graphics metafile (which is sequential), and a GKS Metafile input workstation, which reads from the metafile. In addition to the normal functions for output to workstations, a GKS Metafile output workstation may accept items containing non-graphical information. Input from a metafile is controlled by read and interpret functions which have the same effect as invoking the corresponding functions directly from the application program.

4.4 Graphical output

4.4.1 Output primitives

The graphical information that is generated by GKS and routed to all active workstations is built up of basic pieces called output primitives. GKS provides six output primitives:

- | | |
|--|---|
| a) POLYLINE: | GKS generates a set of connected lines defined by a point sequence. |
| b) POLYMARKER: | GKS generates symbols of one type centred at given positions. |
| c) TEXT: | GKS generates a character string at a given position. |
| d) FILL AREA: | GKS generates a polygonal area which may be hollow or filled with a uniform colour, a pattern, or a hatch style. |
| e) CELL ARRAY: | GKS generates an array of pixels with individual colours. |
| f) GENERALIZED DRAWING PRIMITIVE(GDP): | GKS 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 applies all transformations to the points but leaves the interpretation to the workstation. |

4.4.2 Output primitive attributes

Each output primitive potentially has three types of attribute (geometric, non-geometric and identification). The first two attribute types determine the exact appearance of the output primitive while the third attribute type is used in connection with input. The values of these attributes are set modally and are recorded in the GKS state list. A separate GKS function is provided for each primitive attribute (except the ASFs: see later in this sub-clause), to allow the application program to specify the value of an attribute without unnecessarily specifying the values of other attributes. During creation of an output primitive (that is, when one of the GKS output primitive functions is invoked) these values are bound to the primitive and cannot be changed afterwards.

Attributes of the first type control the geometric aspects of primitives; these are aspects which affect the shape or size of the whole primitive (for example, CHARACTER HEIGHT for TEXT). Hence, they are sometimes referred to as geometric attributes. Attributes of this type are workstation independent and, if they represent coordinate data (points or displacements), are expressed in world coordinates (for example, CHARACTER HEIGHT is expressed in world coordinates but TEXT PATH takes one of a set of enumerated values). They are defined separately for each primitive and a primitive may have zero, one or many geometric attributes.

Current values of (workstation independent) geometric attributes, which are expressed in world coordinates, are stored in world coordinates. When they 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 transformation and the workstation transformation.

Attributes of the second type control the non-geometric aspects of primitives; these are aspects which merely affect a primitive's appearance (for example, linetype for POLYLINE, or colour index for all primitives except CELL ARRAY) or the shape or size of the component parts of the primitive (for example, marker size scale factor for POLYMARKER). Non-geometric aspects do not represent coordinate data. The non-geometric aspects of a primitive may be specified in one of two ways, namely via a bundle or individually.

For specification of aspects via a bundle, there is one attribute per primitive, called the <primitive> INDEX. This attribute is an index into a bundle table, each entry of which contains all the

The Graphical Kernel System

Graphical output

non-geometric aspects of the primitive. There is a separate bundle table for each primitive with the exception of GENERALIZED DRAWING PRIMITIVE and CELL ARRAY (see later in this sub-clause). The non-geometric aspects are workstation dependent in this method of specification and each workstation has its own set of bundle tables (stored in the workstation state list). The values in a particular bundle (or entry in the bundle table) may be different for different workstations.

For individual specification of aspects, there is a separate attribute for each non-geometric aspect. As with the attributes controlling the geometric aspects, these attributes are workstation independent and are stored in the GKS state list. Since each non-geometric aspect only occurs in one primitive bundle type, each of these attributes applies to only one primitive type.

For a given non-geometric aspect, the values that can be assigned to the appropriate bundle component are the same as the values that can be assigned to the corresponding attribute for individual specification. Since the bundles are set separately for each workstation, the values of their components are restricted to the valid values for that workstation. In the case of individual attribute specification, such restrictions are not imposed. Default actions for the display of a primitive are defined to occur if it is created with a value of an individually specified attribute that is invalid on a particular workstation.

As indicated above, GENERALIZED DRAWING PRIMITIVE (GDP) and CELL ARRAY do not have associated bundle tables nor corresponding individually specified attributes. The GDP may use the most appropriate bundle tables or sets of individually specified attributes for each GDP function. For example, if one GDP function is essentially a FILL AREA, then the fill area bundle table or the set of individually specified fill area attributes would be used. CELL ARRAY contains colour index information as part of its definition but has no other non-geometric aspects and so does not use a bundle table nor does it have a set of individually specified attributes.

The method of specification of the non-geometric aspects of a primitive may be chosen separately for each aspect. A further group of primitive attributes, called ASPECT SOURCE FLAGS (ASFs), take the values INDIVIDUAL and BUNDLED to specify the choice. As with the other primitive attributes, these attributes are workstation independent and are stored in the GKS state list. There is one ASF for each non-geometric aspect of each primitive. The initial values of all the ASFs are the same; this may be either BUNDLED or INDIVIDUAL, the choice being implementation dependent. If the initial values are not altered, the system will operate

- a) as if individual specification of non-geometric aspects were not a system feature, if the initial values of all the ASFs are BUNDLED;
- b) as if specification of non-geometric aspects via a bundle were not a system feature, if the initial values of all the ASFs are INDIVIDUAL.

The flags may be set at any time when GKS is open by the function SET ASPECT SOURCE FLAGS. This enables some non-geometric aspects of a primitive to be specified individually and others via a bundle.

When a primitive is displayed, the values of the non-geometric aspects with which it is displayed are determined as follows.

- c) If the ASF for an aspect is INDIVIDUAL, the value used on all workstations is the value of the corresponding individually specified attribute of that primitive.
- d) If the ASF for an aspect is BUNDLED, the value used on a workstation is obtained via the bundle table for that primitive on the workstation; the corresponding component of the bundle, pointed to by the bundle index, is used.

If colour is a non-geometric aspect of a primitive, then it is specified as an index into a separate colour table. There is only one colour table per workstation into which all the colour indices point. Similarly, other entries in a bundle, or corresponding individually specified attributes, may be indices either into another workstation table (for example, style index when interior style PATTERN is used) or into a fixed list (for example, line-types for polyline).

There is precisely one attribute of the third type per primitive, namely PICK IDENTIFIER. This is used for identifying a primitive, or a group of primitives, in a segment when that segment is picked.



The attributes which apply to each output primitive (attributes controlling non-geometric aspects, geometric attributes and PICK IDENTIFIER) are:

- | | | |
|----|-------------------------------------|---|
| e) | POLYLINE: | POLYLINE INDEX
LINETYPE
LINEWIDTH SCALE FACTOR
POLYLINE COLOUR INDEX
LINETYPE ASF
LINEWIDTH SCALE FACTOR ASF
POLYLINE COLOUR INDEX ASF
PICK IDENTIFIER |
| f) | POLYMARKER: | POLYMARKER INDEX
MARKER TYPE
MARKER SIZE SCALE FACTOR
POLYMARKER COLOUR INDEX
MARKER TYPE ASF
MARKER SIZE FACTOR ASF
POLYMARKER COLOUR INDEX ASF
PICK IDENTIFIER |
| g) | TEXT: | TEXT INDEX
TEXT FONT AND PRECISION
CHARACTER EXPANSION FACTOR
CHARACTER SPACING
TEXT COLOUR INDEX
TEXT FONT AND PRECISION ASF
CHARACTER EXPANSION FACTOR ASF
CHARACTER SPACING ASF
TEXT COLOUR INDEX ASF
CHARACTER HEIGHT
CHARACTER UP VECTOR
TEXT PATH
TEXT ALIGNMENT
PICK IDENTIFIER |
| h) | FILL AREA: | FILL AREA INDEX
FILL AREA INTERIOR STYLE
FILL AREA STYLE INDEX
FILL AREA COLOUR INDEX
FILL AREA INTERIOR STYLE ASF
FILL AREA STYLE INDEX ASF
FILL AREA COLOUR INDEX ASF
PATTERN SIZE
PATTERN REFERENCE POINT
PICK IDENTIFIER |
| i) | CELL ARRAY: | PICK IDENTIFIER |
| j) | GENERALIZED DRAW-
ING PRIMITIVE: | Zero or more of the sets e) to i) except that PICK IDENTIFIER is always an attribute |

Figure 2 shows the binding of the attributes.

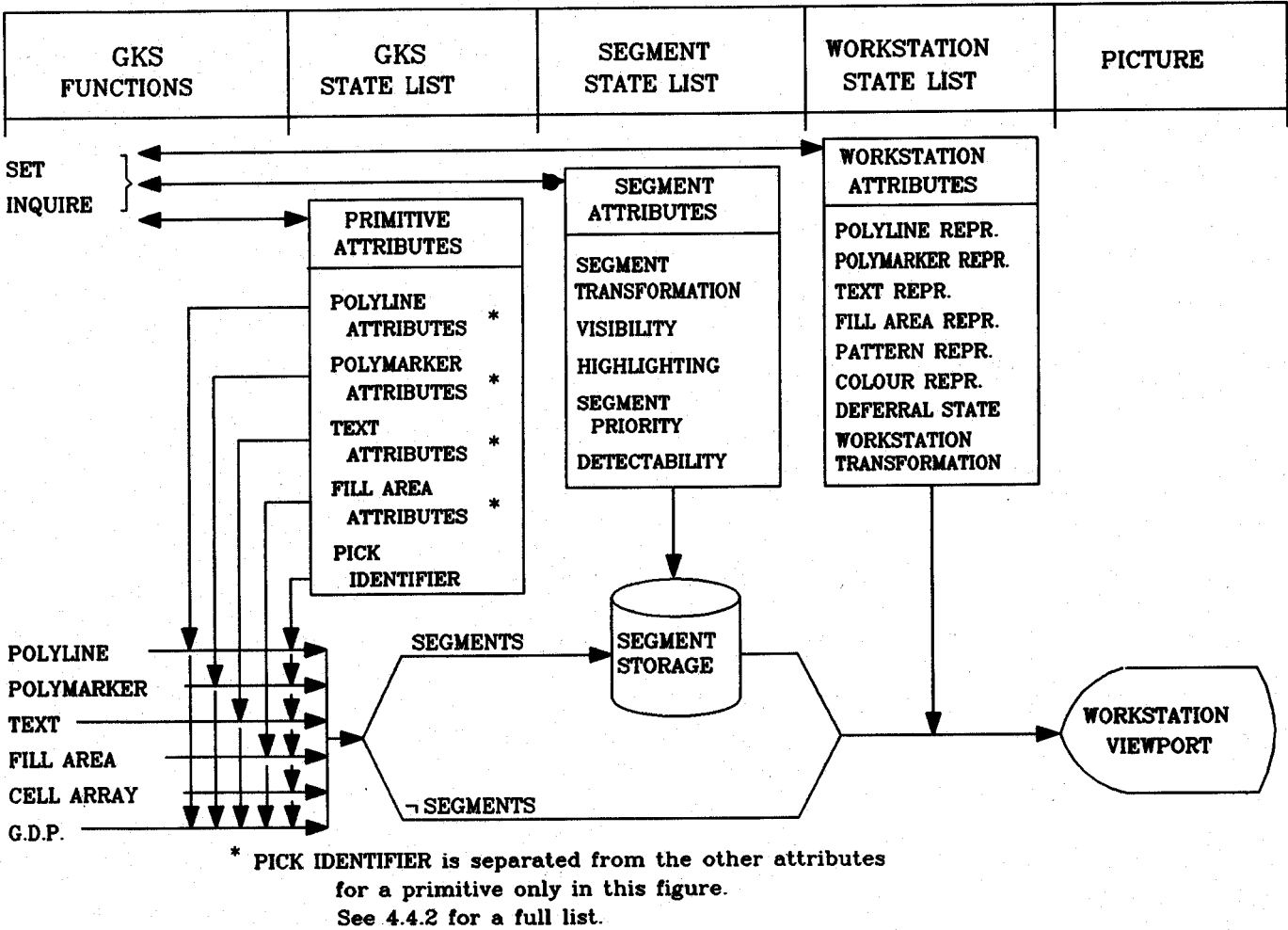


Figure 2 - Binding of attributes

The attributes for each primitive, other than PICK IDENTIFIER, are described in 4.4.3 to 4.4.8. PICK IDENTIFIER is described in more detail in 4.7.1. In the descriptions, attributes appear in upper case (for example, the attributes CHARACTER HEIGHT and PICK IDENTIFIER); aspects appear in both upper and lower case, according to their context. Geometric aspects are always controlled by geometric attributes and so appear in upper case (for example, the aspect CHARACTER HEIGHT). Non-geometric aspects may be controlled via a <primitive> INDEX or by individually specified attributes. Non-geometric aspects appear in lower case unless the corresponding individually specified attributes are being used which appear in upper case (for example, the aspect linetype but the individually specified attribute LINETYPE).

The entries in the bundle, pattern, and colour tables may be set separately for each workstation. Some standard definitions for table 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 values of a specific entry explicitly. Only the most commonly used (or anticipated) combinations of values need be predefined for each output type workstation. At least those predefined entries with indices up to the minimum number of predefined entries at a given level (see 4.10.3) are distinguishable from each other. Other combinations of values can be specified by the SET <primitive | PATTERN | COLOUR> REPRESENTATION function, possibly after inquiring the workstation capabilities. The tables, which are on every workstation of category OUTPUT, OUTIN or MO (i.e. they are workstation attributes), 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 (dynamically) changed. In fact, the only way of changing the aspects of a primitive which are stored in a bundle table is by changing that table. However, note that a change in a bundle table entry can only be reflected in a displayed primitive if the values of the corresponding ASFs (of that primitive) for the aspects in the bundle table are BUNDLED. The entry 'dynamic modification accepted' in the workstation description table indicates which changes

k) lead to an implicit regeneration (may be deferred) (IRG);

l) can be performed immediately (IMM).

The deferral state is explained in more detail in 4.5.3. If changes can be performed immediately, those changes may affect primitives outside segments in addition to those inside segments.

4.4.3 Polyline attributes

Polyline has no geometric attributes. The representation of polyline on the workstation is controlled by the POLYLINE INDEX, or the set of individually specified polyline attributes (LINETYPE, LINEWIDTH SCALE FACTOR, and POLYLINE COLOUR INDEX) or some combination of the two, depending upon the values of the ASFs for linetype, linewidth scale factor and polyline colour index. The POLYLINE INDEX is a pointer into the polyline bundle table, each entry of which contains values for linetype, linewidth scale factor and polyline colour index.

Linetypes 1 to 4 are solid, dashed, dotted and dashed-dotted. Every workstation of category OUTPUT or OUTIN realizes linetypes 1 to 4 with recognizable styles. Linetypes greater than 4 are reserved for registration (see 4.1.2). Linetypes less than 0 may be available but their styles are implementation dependent. The linetype specifies a sequence of line segments and gaps which are repeated to draw a 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.

4.4.4 Polymarker attributes

Polymarker has no geometric attributes. The representation of polymarker at the workstation is controlled by the POLYMARKER INDEX, or the set of individually specified polymarker attributes (MARKER TYPE, MARKER SIZE SCALE FACTOR, and POLYMARKER COLOUR INDEX) or some combination of the two, depending upon the values of the ASFs for marker type, marker size scale factor, and polymarker colour index. The POLYMARKER INDEX is a pointer into the polymarker bundle table, each entry of which contains values for marker type, marker size 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. Every workstation of category OUTPUT or OUTIN realizes marker types 1 to 5 with recognizable shapes at the given positions. Marker types greater than 5 are reserved for registration (see 4.1.2). Marker types less than 0 may be available but their forms are implementation dependent.

The marker size is calculated as a nominal size multiplied by the marker size 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.

4.4.5 Text attributes

Text has the geometric attributes CHARACTER HEIGHT, CHARACTER UP VECTOR, TEXT PATH, and TEXT ALIGNMENT which are specified and used as described in this sub-clause.

Text also has two implicitly specified geometric attributes CHARACTER WIDTH and CHARACTER BASE VECTOR. These are implicitly specified by the functions SET CHARACTER HEIGHT and SET CHARACTER UP VECTOR respectively. They otherwise behave like ordinary geometric attributes (their values are bound to TEXT primitives when the primitives are created and cannot be changed afterwards and these values are subject to the same transformations as the geometric data contained in the definition of the primitive).

The representation of text at the workstation is controlled by the TEXT INDEX, or the set of individually specified text attributes (TEXT FONT AND PRECISION, CHARACTER EXPANSION FACTOR, CHARACTER SPACING, and TEXT COLOUR INDEX) or some combination of the two, depending upon the values of the ASFs for text font and precision, character expansion factor, character spacing and text colour index. The TEXT INDEX is a pointer into the text bundle table, each entry of which contains values for text font and precision, character expansion factor, character spacing and text colour index.

Precise control of the appearance of TEXT on a workstation is provided by the following aspects: CHARACTER HEIGHT, CHARACTER WIDTH, character expansion factor, TEXT PATH, CHARACTER UP VECTOR, CHARACTER BASE VECTOR, character spacing and TEXT ALIGNMENT. However, the use of these values in displaying text is determined by the setting of the text font and precision aspect (font and precision are two components of the same aspect). The CHARACTER HEIGHT specifies the nominal height of a capital letter character. The CHARACTER WIDTH specifies the nominal width of a character; the actual width depends upon the width to height ratio of the character indicated by the font designer and may vary from character to character. The character expansion factor specifies the deviation of the width to height ratio of the character from the ratio indicated by the font designer. The CHARACTER UP VECTOR gives the up direction of a character. The CHARACTER BASE VECTOR gives the direction of the baseline of a character. Only the directions, not the lengths, of these vectors are relevant. 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 along a baseline in the direction of the CHARACTER BASE VECTOR. For LEFT, the baseline direction is the opposite direction of the CHARACTER BASE VECTOR. For UP, the character path coincides with the direction of the CHARACTER UP VECTOR. For DOWN, it is the opposite direction of 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.

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 aspects CHARACTER HEIGHT, CHARACTER WIDTH, character expansion factor, TEXT PATH, character spacing and text font 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 and CHARACTER BASE VECTOR attributes 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 this text extent parallelogram in relation to the text position. For simplicity the TEXT ALIGNMENT is described in terms of the default CHARACTER UP VECTOR and CHARACTER BASE VECTOR, 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 3). 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 CHARACTER BASE VECTOR 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)

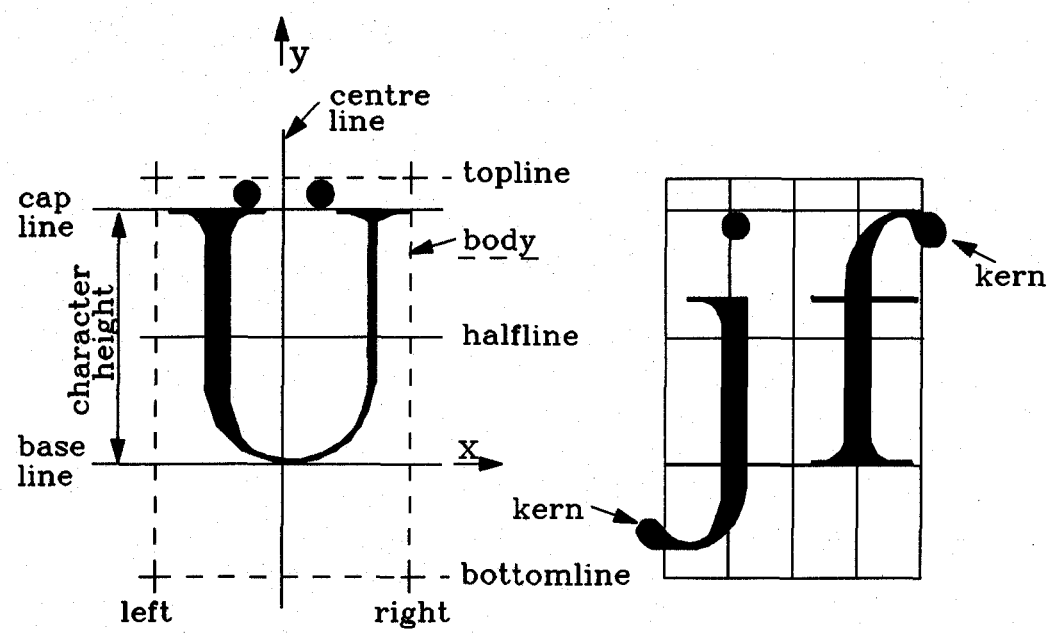


Figure 3 - Font description coordinate system

The initial values of the geometric 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)

and the initial values of the implicitly specified geometric text attributes are:

CHARACTER WIDTH	WC	0.01 (i.e. the same value as the initial value of CHARACTER HEIGHT)
CHARACTER BASE VECTOR	WC	(1,0)

Text font and precision together constitute one aspect. 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 (see 4.1.2). 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 workstation independent 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, CHARACTER WIDTH and character expansion factor are evaluated as closely as reasonable, given the capabilities of the workstation. CHARACTER UP VECTOR, CHARACTER BASE 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 aspects CHARACTER HEIGHT, CHARACTER WIDTH, the up direction of the CHARACTER UP VECTOR, the baseline direction of the CHARACTER BASE VECTOR, 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 aspects 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 aspects. 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.

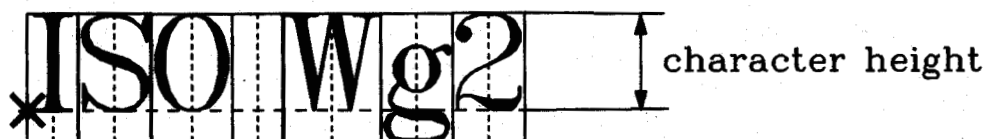
A GKS output level 0 implementation supports text precisions STRING and CHAR. Above output level 0, all text precisions are supported as follows. A workstation may use a higher precision than the one requested for this purpose i.e. if STROKE precision is supported in a particular font, the implication 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. That is, for any GKS level supporting a STROKE precision font, every workstation of a particular installation supports 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 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').

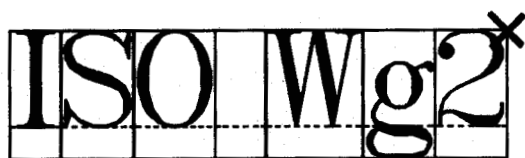
Since the values of CHARACTER HEIGHT, CHARACTER WIDTH, CHARACTER UP VECTOR and CHARACTER BASE VECTOR are given in world coordinate units, but the characters are generated on the workstation in device coordinates, using the workstation dependent font and precision, the geometric attributes need to be transformed in such a way that the workstation can generate the characters in the way intended.



CHARACTER HEIGHT = 1, CHARACTER UP VECTOR = (0,1),
 TEXT PATH = RIGHT, TEXT ALIGNMENT = (NORMAL,NORMAL)



CHARACTER HEIGHT = 0.5, CHARACTER UP VECTOR = (0,1)
 TEXT PATH = RIGHT, TEXT ALIGNMENT = (NORMAL,NORMAL)

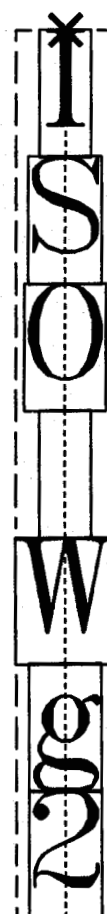


CHARACTER HEIGHT = 1, CHARACTER UP VECTOR = (0,1),
 TEXT PATH = RIGHT, TEXT ALIGNMENT = (RIGHT,TOP)



CHARACTER HEIGHT = 1
 CHARACTER UP VECTOR = (-1,0)
 TEXT PATH = RIGHT
 TEXT ALIGNMENT = (NORMAL,NORMAL)

CHARACTER HEIGHT = 1
 CHARACTER UP VECTOR = (0,1)
 TEXT PATH = DOWN
 TEXT ALIGNMENT = (NORMAL,NORMAL)



× text position

-----baseline or centreline

---text extent rectangle
 (indicated for PATH = DOWN)

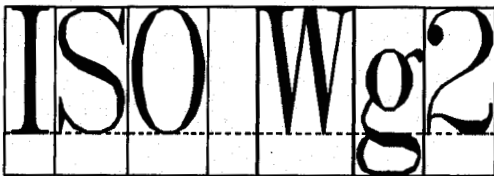
NOTES

- 1 Examples are illustrated with STROKE precision, a character expansion factor of 1 and a zero character spacing.
- 2 Capline = topline in these examples.
- 3 Changed attributes are underlined.

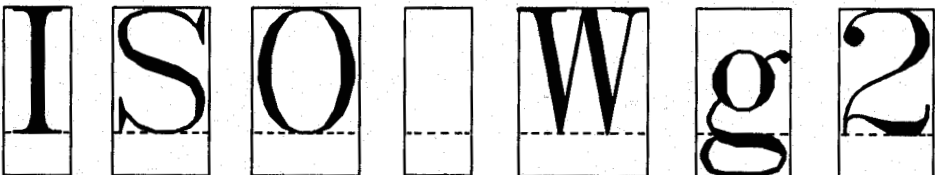
Figure 4 - Effects of changes in geometric text attributes



CHARACTER EXPANSION FACTOR = 1, CHARACTER SPACING = 0



CHARACTER EXPANSION FACTOR = 0.75, CHARACTER SPACING = 0

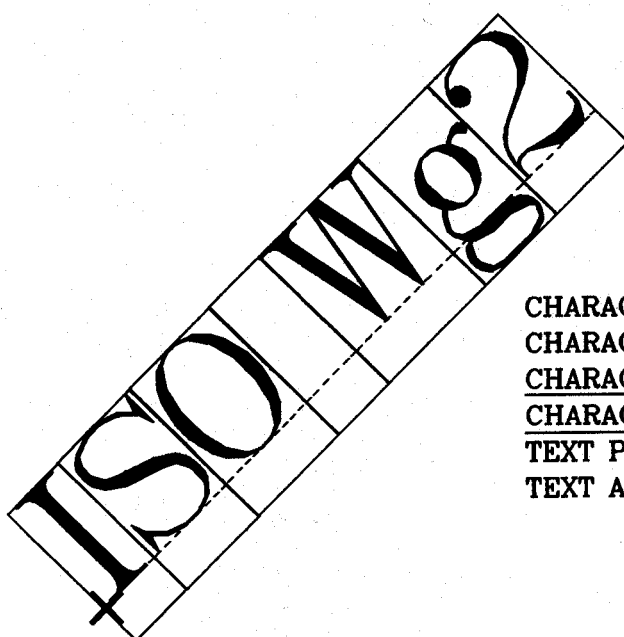


CHARACTER EXPANSION FACTOR = 1, CHARACTER SPACING = 0.3

NOTES

- 1 Examples are illustrated with default values of the geometric text attributes and with STROKE precision.
- 2 Changed aspects are underlined.

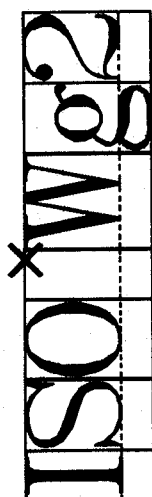
Figure 5 - Effects of changes in non-geometric text aspects



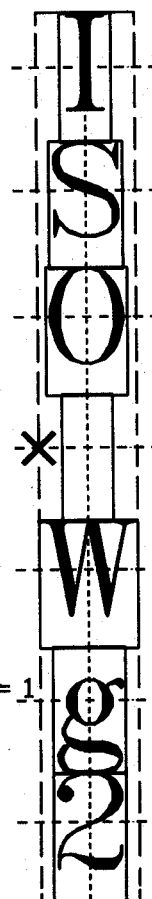
CHARACTER EXPANSION FACTOR = 1
 CHARACTER SPACING = 0
CHARACTER HEIGHT = 1.414
CHARACTER UP VECTOR = (-1,1)
 TEXT PATH = RIGHT
 TEXT ALIGNMENT = (NORMAL,NORMAL)



CHARACTER EXPANSION FACTOR = 1
CHARACTER SPACING = -0.3
 CHARACTER HEIGHT = 1
 CHARACTER UP VECTOR = (0,1)
TEXT PATH = LEFT
 TEXT ALIGNMENT = (NORMAL,NORMAL)



CHARACTER EXPANSION FACTOR = 1
 CHARACTER SPACING = 0
 CHARACTER HEIGHT = 1
CHARACTER UP VECTOR = (-1,0)
 TEXT PATH = RIGHT
TEXT ALIGNMENT = (CENTRE,TOP)

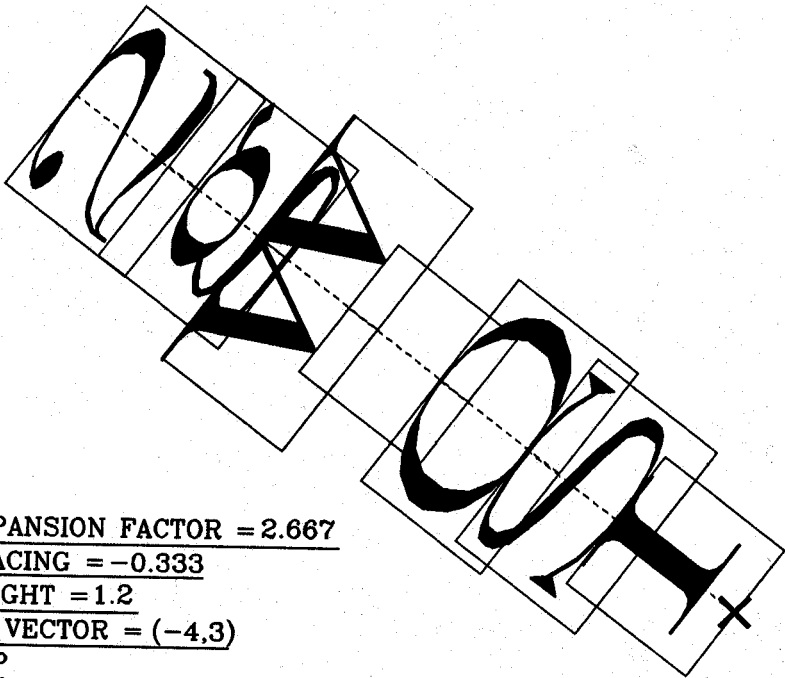


CHARACTER EXPANSION FACTOR = 1
 CHARACTER SPACING = 0
 CHARACTER HEIGHT = 1
 CHARACTER UP VECTOR = (0,1)
TEXT PATH = DOWN
TEXT ALIGNMENT = (LEFT,HALF)

NOTES

- 1 Changes from the top example of figures 4 and 5 are underlined.
- 2 In the last example, halflines of all characters are shown.

Figure 6 - Effects of combined changes in text aspects



CHARACTER EXPANSION FACTOR = 2.667
CHARACTER SPACING = -0.333
CHARACTER HEIGHT = 1.2
CHARACTER UP VECTOR = (-4,3)
TEXT PATH = UP
TEXT ALIGNMENT = (CENTRE,BOTTOM)

NOTE - Changes from the top example of figures 4 and 5 are underlined.

Figure 7 - Effects of several changes in text aspects

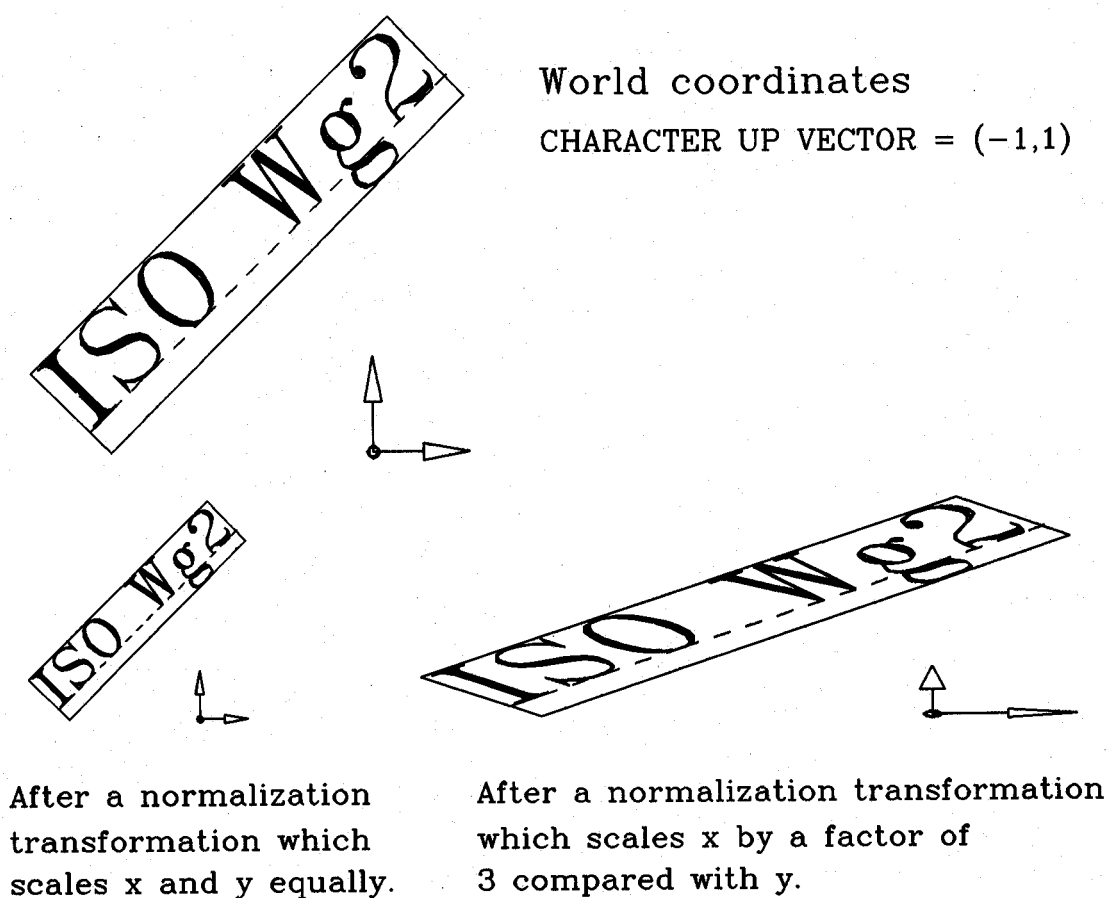


Figure 8 - Effects of different normalization transformations on text in STROKE precision

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 CHARACTER BASE VECTOR with length equal to CHARACTER WIDTH, are passed down the viewing pipeline. These vectors are transformed by the normalization transformation, by a segment transformation if within a segment, and by the workstation transformation. They are also stored in segments. 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 segment 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 4 to 7 give examples of the effects of different values of text aspects. Figure 8 gives examples of the effect of different normalization transformations on the displayed form of the text.

4.4.6 Fill area attributes

Fill area has the geometric attribute PATTERN REFERENCE POINT. It also has two implicitly specified geometric attributes PATTERN WIDTH VECTOR and PATTERN HEIGHT VECTOR. These are implicitly specified by the function SET PATTERN SIZE. Like ordinary geometric attributes, their values are bound to FILL AREA primitives when the primitives are created and cannot be changed afterwards and 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 geometric attributes is described later in this sub-clause.

The representation of fill area at the workstation is controlled by the FILL AREA INDEX, or the set of individually specified fill area attributes (FILL AREA INTERIOR STYLE, FILL AREA STYLE INDEX, and FILL AREA COLOUR INDEX) or some combination of the two, depending upon the values of the ASFs for fill area interior style, fill area style index, and fill area colour index. The FILL AREA INDEX is a pointer into the fill area bundle table, each entry of which contains values for the 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 (either via the fill area bundle or individually, depending upon the corresponding ASF). The linetype and linewidth are implementation dependent.
- b) **SOLID:** Fill the interior of the polygon using the fill area colour index currently selected (either via the fill area bundle or individually, depending upon the corresponding ASF).
- c) **PATTERN:** Fill the interior of the polygon using the fill area style index currently selected (either via the fill area bundle or individually, depending upon the corresponding ASF) 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 (either via the fill area bundle or individually, depending upon the corresponding ASFs). 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

The Graphical Kernel System

Graphical output

REFERENCE POINT. The pattern box is conceptually divided into a grid of $DX \times DY$ 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: hatch styles greater than 0 are reserved for registration (see 4.1.2); hatch styles less than 0 are workstation dependent. Whether hatching is affected by transformations or not is workstation dependent.

Interior style **HOLLOW** is available on every workstation of category **OUTPUT** or **OUTIN**. It is workstation dependent which of the interior styles **SOLID**, **PATTERN** and **HATCH** are available.

4.4.7 Cell array attributes

Cell array has no attributes other than **PICK IDENTIFIER**. However, an array of colour indices, which are pointers into the colour table, is part of the definition of a cell array.

4.4.8 Generalized Drawing Primitive attributes

Generalized Drawing Primitive(GDP) has no explicit geometric attributes. Such information may be specified in the GDP data record. The representation of the GDP at the workstation is controlled by zero or more of the sets of polyline, polymarker, text and fill area attributes (see 4.4.2). Whether bundle indices or associated individually specified attributes are used depends upon the values of the appropriate ASFs. The sets of attributes most appropriate for the specified GDP function are selected for the GDP as part of the definition of the GDP and are recorded in the workstation description table. Similarly, if a GDP is essentially a cell array, then an array of colour indices would be specified in the GDP data record.

4.4.9 Colour

In GKS, colour is specified in a number of different situations. It may be an aspect of a primitive, in which case it is specified either in the bundle for that primitive or by the individual colour attribute for that primitive. It may be part of a pattern for **FILL AREA**, in which case an array of colours is specified, or it may be part of a primitive itself, namely **CELL ARRAY**, when an array of colours 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 surface after it has been cleared. Entry 1 is the default foreground colour and entries higher than 1 correspond to alternative foreground colours. Entries in the table may be set by the function **SET COLOUR REPRESENTATION** which specifies the colour as a combination of red, green, and blue intensities. The specified colour is mapped to the nearest available by 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.

4.5 Workstations

4.5.1 Workstation characteristics

GKS is based on the concept of abstract graphical workstations. These provide the logical interface through which the application program controls physical devices. Certain special workstations provide facilities for the storage and exchange of graphical information.

For every type of workstation present in a GKS implementation (except for the special workstations), there exists a workstation description table which describes the capabilities and characteristics of the workstation. The application program can inquire which capabilities are available and adapt its behaviour accordingly. If a capability is requested that a particular workstation does not provide, a standard error reaction is defined. Certain minimal capabilities of a workstation are detailed in 4.10.

An abstract graphical workstation with maximum capabilities

- a) has one addressable display surface 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) supports several linetypes, text fonts, character sizes, etc., to allow output primitives to be drawn with different aspects;
- 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;
- h) stores segments and provides facilities for changing and manipulating them.

In practice, the workstation is not necessarily equipped with all of these capabilities.

Each workstation has a type. Each workstation type falls into one of six categories:

OUTPUT	Output
INPUT	Input
OUTIN	Output and input
WISS	Workstation Independent Segment Storage(WISS)
MO	GKS Metafile (GKSM) output
MI	GKSM input

A workstation of category OUTPUT has only output capabilities. It can display all output primitives, with the possible exception of the GENERALIZED DRAWING PRIMITIVE which is optional. Minimal requirements for displaying TEXT and FILL AREA primitives are listed in 4.4, and for CELL ARRAY in 5.3.

GKS allows the appearance of output primitives to vary between workstations, thus allowing advantage to be taken of their differing capabilities. The facilities which allow this variation are:

- polyline representation (see 4.4)
- polymarker representation (see 4.4)
- text representation (see 4.4)
- fill area representation (see 4.4)
- pattern representation (see 4.4)
- colour representation (see 4.4)
- deferral state (see 4.5.3)
- workstation transformation (see 4.6.3)

Figure 2 (see 4.4) illustrates the binding of the workstation attributes.

A workstation of category INPUT has at least one logical input device, but no output capability.

A workstation of category OUTIN has the characteristics of both an OUTPUT and INPUT workstation. In addition, the existence of a workstation in this category in a GKS implementation gives rise to additional requirements regarding logical input devices (see 4.8.1).

The last three categories WISS, MO and MI are special GKS facilities that provide a means for temporarily or permanently storing graphical information. They are treated as workstations for the purposes of control, but otherwise have quite different characteristics (see 4.7.5 and 4.9).

Clause A.5 of annex A gives a complete listing of all GKS functions which apply directly or indirectly to each category of workstation.

Actual workstations may provide more capabilities than those listed in the workstation description table. These cannot be used by GKS. 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 or the application program. More than one display surface can be controlled by GKS only by defining a separate workstation for each display surface.

4.5.2 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 and a connection identifier (for example, a unit number in FORTRAN). The current state of each open workstation is kept in a workstation state list. Segment manipulation and input can be performed on all open workstations. Output primitives are sent to, and segments are stored on, all active workstations and no others; an open workstation is made active by the function ACTIVATE WORKSTATION.

An active workstation is made inactive by the function DEACTIVATE WORKSTATION; an open workstation is closed by the function CLOSE WORKSTATION.

The following sequence of functions illustrates workstation selection:

```

OPEN WORKSTATION (N1,conid1,workstation type A);
OPEN WORKSTATION (N2,conid2,workstation type B);
ACTIVATE WORKSTATION (N1);

    Output functions;           {generated only on N1}
    Input functions;           {possible on N1,N2}

ACTIVATE WORKSTATION (N2);

    Output functions;           {generated on N1,N2}
```

DEACTIVATE WORKSTATION (N1);

Output functions; {generated only on N2}
Input functions; {possible on N1,N2}

CLOSE WORKSTATION (N1);
DEACTIVATE WORKSTATION (N2);
CLOSE WORKSTATION (N2);

4.5.3 Deferring picture changes

It is desirable that the display of a workstation reflects, as far as possible, the actual state of the picture as defined by the application program. However, to use the capabilities of a workstation efficiently, GKS allows a workstation to delay, for a certain period of time, the actions requested by the application program. During this period, the state of the display may be undefined.

The function SET DEFERRAL STATE allows the application program to choose that deferral state which takes into account the capabilities of the workstation and the requirements of the application program. Two attributes are defined for this purpose. Deferral mode controls the time at which output functions have their visual effects. Implicit regeneration controls the time at which picture changes have their visual effects: picture changes in general imply an alteration not just an addition to the picture.

The concept of deferral refers only to visible effects of GKS functions. Effects on the segment storage or on the state of the workstation are (conceptually) not deferred.

Deferral mode controls the possible delaying of output functions: for example, data sent to a device may be buffered to optimize data transfer. The values of deferral mode (in increasing order of delay) are:

- a) ASAP: The visual effect of each function will be achieved on the workstation As Soon As Possible (ASAP). GKS ensures that the actions necessary to achieve this visual effect are initiated before control is returned to the application program, but, owing to possible delays not under the influence of GKS, the actions are not necessarily completed before control is returned.
- b) BNIG: The visual effect of each function will be achieved on the workstation Before the Next Interaction Globally (BNIG), i.e. before the next interaction with a logical input device gets underway on any workstation (see 4.8.2). If an interaction on any workstation is already underway, the visual effect will be achieved as soon as possible.
- c) BNIL: The visual effect of each function will be achieved on the workstation Before the Next Interaction Locally (BNIL), i.e. before the next interaction with a logical input device gets underway on that workstation (see 4.8.2). If an interaction on that workstation is already underway, the visual effect will be achieved as soon as possible.
- d) ASTI: The visual effect of each function will be achieved on the workstation At Some Time (ASTI).

Deferral applies to the following functions that generate output:

POLYLINE
POLYMARKER
TEXT
FILL AREA
CELL ARRAY
GENERALIZED DRAWING PRIMITIVE
INSERT SEGMENT
ASSOCIATE SEGMENT WITH WORKSTATION
COPY SEGMENT TO WORKSTATION
INTERPRET ITEM

For none of the possible values of deferral mode is it mandatory for an implementation to delay the visual

effect of output functions. If the application program requires a delay, it can achieve this using the segment storage facility and the visibility attribute. This restriction means that the buffer for deferred actions can be chosen in an implementation dependent manner.

Certain functions can be performed immediately on some workstations, but on other workstations imply a regeneration of the whole picture to achieve their effect. For example, an implicit regeneration is necessary when picture changes require new paper to be put on a plotter. The entries 'dynamic modification accepted' in the workstation description table indicate which changes

e) lead to an implicit regeneration (IRG);

f) can be performed immediately (IMM).

If changes can be performed immediately, those changes may affect primitives outside segments in addition to those inside segments. If regeneration occurs, all primitives outside segments will be deleted from the display surface.

An implicit regeneration is equivalent to an invocation of the function REDRAW ALL SEGMENTS. Its possible delay is controlled by the implicit regeneration mode, a single entry in the workstation state list. The values of implicit regeneration mode are

g) **SUPPRESSED:** Implicit regeneration of the picture is suppressed, until it is explicitly requested: the entry 'new frame necessary at update' is set to YES.

h) **ALLOWED:** Implicit regeneration of the picture is allowed.

An implicit regeneration is made necessary, if any of the following occur:

i) if the functions listed below have a visible effect on the display image of the respective workstation:

1) if the 'dynamic modification accepted' entry in the workstation description table is IRG (implicit regeneration necessary) for the specified representation:

SET POLYLINE REPRESENTATION
SET POLYMARKER REPRESENTATION
SET TEXT REPRESENTATION
SET FILL AREA REPRESENTATION
SET PATTERN REPRESENTATION
SET COLOUR REPRESENTATION

2) if the 'dynamic modification accepted' entry in the workstation description table is IRG for the workstation transformation:

SET WORKSTATION WINDOW
SET WORKSTATION VIEWPORT

3) if the 'dynamic modification accepted' entry in the workstation description table is IRG for segment priority and this workstation supports segment priority:

- if primitives are added to open segment overlapping a segment of higher priority:

POLYLINE
POLYMARKER
TEXT
FILL AREA
CELL ARRAY
GENERALIZED DRAWING PRIMITIVE
INSERT SEGMENT

(since only segments have priority, primitives outside segments do not make an implicit regeneration necessary.)

- if the complete execution of one of the following actions would be affected by segment priority:

DELETE SEGMENT
 DELETE SEGMENT FROM WORKSTATION
 ASSOCIATE SEGMENT WITH WORKSTATION
 SET SEGMENT TRANSFORMATION
 SET VISIBILITY
 SET SEGMENT PRIORITY

4) if the 'dynamic modification accepted' entry in the workstation description table is IRG for segment transformation:

SET SEGMENT TRANSFORMATION

5) if the 'dynamic modification accepted' entry in the workstation description table is IRG for 'visibility (visible → invisible)':

SET VISIBILITY (INVISIBLE)

6) if the 'dynamic modification accepted' entry in the workstation description table is IRG for 'visibility (invisible → visible)':

SET VISIBILITY (VISIBLE)

7) if the 'dynamic modification accepted' entry in the workstation description table is IRG for highlighting:

SET HIGHLIGHTING

8) if the 'dynamic modification accepted' entry in the workstation description table is IRG for delete segment:

DELETE SEGMENT
 DELETE SEGMENT FROM WORKSTATION

j) if any of the above situations occurs as a result of INTERPRET ITEM.

An implicit regeneration has to be done (including deletion of primitives outside segments) only if one of the functions listed causes a visible effect on the display; for example, if an invisible segment is deleted, a regeneration need not be done. However, an implementation is allowed to perform an implicit regeneration in any of the cases listed above.

Deferred actions can be made visible at any time by the use of the UPDATE WORKSTATION function or by an appropriate change of the deferral state.

4.5.4 Clearing the display surface

Two capabilities for clearing the display surface are recognized, namely:

- a) clear the display surface even if it is empty;
- b) ensure that the display surface is clear without clearing the display surface needlessly.

The second capability means that the display surface is only cleared when needed: this would normally be when the display surface is not clear, (i.e. when the 'display surface empty' entry in the workstation state list is NOTEMPTY). The entry 'display surface empty' in the workstation state list is set to NOTEMPTY if output is sent to the device. It may be set to NOTEMPTY even if output does not appear on the display surface (for example, a GDP primitive which has been clipped at the device to non-existence).

Both capabilities for clearing the display surface are available to the user through the function CLEAR WORKSTATION. The second capability is also used in UPDATE WORKSTATION and REDRAW ALL SEGMENTS ON WORKSTATION.

4.5.5 Elimination of primitives outside segments

Elimination of primitives outside segments occurs in the following situations:

- a) if the following GKS functions are invoked:

CLEAR WORKSTATION

REDRAW ALL SEGMENTS ON WORKSTATION

UPDATE WORKSTATION

if the parameter update regeneration flag is **PERFORM** and if 'new frame action necessary at update' in the workstation state list is **YES**;

SET DEFERRAL STATE

if 'implicit regeneration mode' is **ALLOWED** and 'new frame action necessary at update' is **YES**;

- b) if an implicit regeneration is made necessary (see 4.5.3) and 'implicit regeneration mode' is **ALLOWED**;
- c) if any of the above situations occurs as a result of **INTERPRET ITEM**.

4.5.6 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. The rules to be followed by an implementation are stated in 5.2.

4.6 Coordinate systems and transformations

4.6.1 Normalization transformations

In GKS, the application programmer can compose his graphical picture from separate parts each of which, conceptually, is defined with its own world coordinate system (WC). The relative positioning of the separate parts 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 surface. This normalized picture can be stored and manipulated via the segment mechanism; it can also be stored on a metafile.

For output, a single normalization transformation is current at any one time and this is used to transform world coordinates specified, for example in output primitives and geometric attributes, into normalized device coordinates.

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. 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 part of NDC space in which the viewport needs to be located and that can be viewed at a workstation is the closed range $[0,1] \times [0,1]$. In addition, an implementation may support only a restricted range of NDCs. However, this range is always sufficiently greater than the $[0,1] \times [0,1]$ square that useful effects of INSERT SEGMENT can be achieved. In particular, NDCs in the range $[-7,7] \times [-7,7]$ are always handled.

Each normalization transformation is identified by a transformation number which is an integer between 0 and an implementation dependent value n which can be inquired from the GKS state list. The normalization transformation with transformation number 0 is the unity transformation which maps $[0,1] \times [0,1]$ in world coordinates to $[0,1] \times [0,1]$ 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 is open. Since GKS 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 parts of 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.

4.6.2 Clipping

The viewport of a particular normalization transformation is used to define a clipping rectangle, as well as, with the window, specifying the normalization transformation. When the viewport of the current normalization transformation is set or when a normalization transformation is selected, the 'clipping rectangle' entry in the GKS state list is set to the resulting viewport of the current normalization transformation. Clipping to this clipping rectangle can either be enabled or disabled. There is a single global switch (the clipping indicator) which defines whether or not the clipping rectangle is to be used for clipping.

Clipping does not take place when the normalization transformation is performed but is delayed until the output primitives are to be displayed on the display surface of a workstation. Output primitives stored in segments have their coordinates transformed to NDC and the associated clipping rectangle is stored with the primitives. The INSERT SEGMENT function allows the clipping rectangle in the GKS state list to replace the clipping rectangle, stored with an output primitive when the segment was defined.

Primitives sent to a workstation of category MO are not clipped.

4.6.3 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 $[0,1] \times [0,1]$ to be displayed somewhere on the workstation display surface. 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 size of the display space in device coordinate units is recorded in the workstation description table.

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.

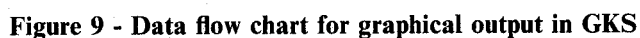
The workstation transformation can be specified at any time after the workstation has been opened. However, actually changing the workstation transformation may cause an implicit regeneration of the picture.

A workstation transformation is specified by defining the limits of an area in the normalized device coordinate system within the range $[0,1] \times [0,1]$ (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 $[0,1] \times [0,1]$, this ensures that the only part of NDC space that can be viewed on any workstation lies in the range $[0,1] \times [0,1]$. 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 $[0,1] \times [0,1]$ 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 can be changed by SET WORKSTATION WINDOW or SET WORKSTATION VIEWPORT. As a specification of the workstation transformation may be deferred (see 4.5.3), these two functions only set the 'requested workstation window' and 'requested workstation viewport' entries in the workstation state list. The 'current workstation window' and 'current workstation viewport' entries continue to hold the previously set transformation parameters. When the display is updated, the current values are set to the requested values.

A complete data flow chart for graphical output is given in figure 9. It should be noted that all three coordinate systems (WC, NDC and DC) are two dimensional Cartesian coordinate systems.

4.6.4 Transformation of locator 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 $[0,1] \times [0,1]$.

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 initialisation, an implementation defined number of normalization transformations are initialised 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.

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.

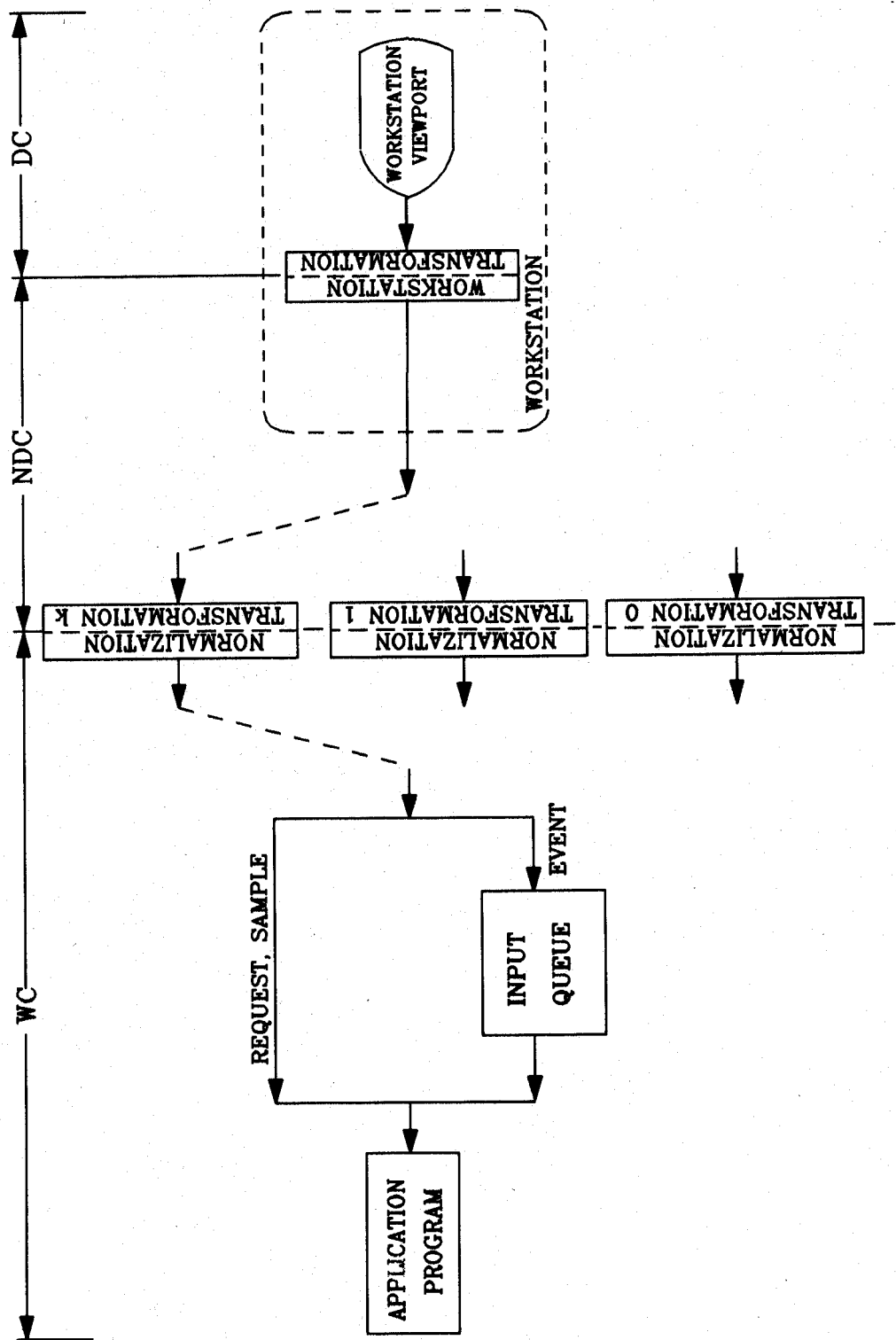


Figure 10 - Data flow chart for locator input

Between placing on the input queue and the execution of **AWAIT EVENT** which removes the **LOCATOR** position 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 position located is equivalent to the WC position retrieved from the input queue, it is advisable for the application program not to change a transformation while a **LOCATOR** is in **EVENT** mode.

4.6.5 Transformation of stroke input

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 $[0,1] \times [0,1]$.

To return to the application program points in world coordinates, the points in NDC space need to be transformed from NDC to WC by the inverse of one of the normalization transformations. 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.

4.7 Segments

4.7.1 Introduction to segments

In GKS, graphical output primitives may be grouped in segments as well as being created outside segments. Each segment is identified by a unique, application specified segment name. Segments may be

- a) transformed;
- b) made visible or invisible;
- c) highlighted;
- d) ordered front to back, which impacts overlapping primitives;
- e) made detectable or undetectable;
- f) deleted;
- g) renamed;
- h) inserted into the open segment or into the stream of primitives outside segments (see 4.7.6).

Only primitives contained inside segments are affected by these operations. The application program has no access to primitives outside segments once they have been created.

Every primitive within a segment has an attribute **PICK IDENTIFIER** which establishes a second level of naming. The sole function of the **PICK IDENTIFIER** is the identification of primitives; it cannot be used for manipulations. This level of naming is provided in GKS to reduce the segment overhead for applications where a great number of picture parts need to be distinguished for input but the need for manipulation is less important.

Whereas segment names are unique, the same value for **PICK IDENTIFIER** can be assigned arbitrarily to single output primitives or groups of output primitives within segments, as illustrated in the following sequence of functions:

```

SET PICK IDENTIFIER(4);
CREATE SEGMENT (1);
  Output functions;           {segment = 1, PICK IDENTIFIER = 4}
SET PICK IDENTIFIER (2);
  Output functions;           {segment = 1, PICK IDENTIFIER = 2}
CLOSE SEGMENT;
  Output functions;           {primitives not pickable}
                              {PICK IDENTIFIER = 2}

SET PICK IDENTIFIER (5);
  Output functions;           {primitives not pickable}
                              {PICK IDENTIFIER = 5}

CREATE SEGMENT (2);
  Output functions;           {segment = 2, PICK IDENTIFIER = 5}
SET PICK IDENTIFIER (3);
  Output functions;           {segment = 2, PICK IDENTIFIER = 3}
CLOSE SEGMENT;
```

After a segment is closed, primitives in it cannot be modified nor can primitives be added to or deleted from the segment. No function is provided to extend a segment after it has been closed. Clipping rectangles and primitive attributes (geometric attributes, attributes controlling non-geometric aspects, and **PICK IDENTIFIER**) are stored along with primitives in segment storage. Geometrical transformations, changes of the segment attributes and changes of the workstation specific bundle and colour tables referenced from within a segment are possible. All values describing the state of a segment (i.e. name, segment attributes, and workstations active at creation time) are stored in a segment state list that GKS keeps during a segment's lifetime.

Each segment is stored on all workstations active at the time the segment is created (**CREATE SEGMENT**). It can be deleted on all workstations by the **DELETE SEGMENT** function. It can be deleted on a specific workstation by the **DELETE SEGMENT FROM WORKSTATION** function. All segments stored on a

specific workstation can be deleted from it by the CLEAR WORKSTATION function.

Segment storage on an OUTPUT or OUTIN workstation is referred to as Workstation Dependent Segment Storage (WDSS). GKS provides a second segment storage system for workstation independent storage of pictures at run time, Workstation Independent Segment Storage (WISS: see 4.7.5 and 4.7.6). Segments cannot be moved from WDSS to another workstation, but from WISS they can.

Segments have a unique name across all segment storage. A GKS implementation provides a large number of available segment names (for example 32 000).

4.7.2 Segment attributes

Segment attributes affect all the primitives in a segment. The segment attributes are:

- a) SEGMENT TRANSFORMATION: (see 4.7.3);
- b) VISIBILITY: a segment is either displayed or not;
- c) HIGHLIGHTING: a visible segment is either highlighted or not;
- d) SEGMENT PRIORITY: if parts of segments (for example, FILL AREA, CELL ARRAY) overlap, the segment with the higher priority will be preferred, both when the segments are displayed and when they are picked;
- e) DETECTABILITY: a segment can either be selected by a pick input device or it cannot.

The segment attributes are unique for each segment and do not vary on different workstations. The default segment attributes (identity transformation, visible, not highlighted, priority 0, undetectable) are assigned to a segment when it is opened. The segment attributes of any segment in existence, including the open segment, may be changed. The binding of segment attributes is shown in figure 2 (see 4.4).

Segment priority affects segments being displayed (i.e. performing segment and workstation transformations, including clipping, for each primitive of the segment). If parts of primitives overlap with others of a visible segment with higher priority, these parts may be invisible. Whether a workstation supports this feature is indicated in the workstation description table. This feature is intended to address appropriate hardware capabilities only. It is not intended to mandate shielding on non-raster displays. When primitives within a segment overlap, the implementation determines the appearance of the overlapped parts. The actual effect is listed in the documentation accompanying an implementation.

When primitives of segments overlapping each other are picked, the segment with higher priority is selected. When primitives of the same segment or of segments with equal priority overlap, the results are implementation dependent.

4.7.3 Segment transformations

Segment transformations are a mapping from NDC onto NDC. They perform translation, scaling and rotation.

Segment transformations are characterized by:

- a) segment name;
- b) transformation matrix.

The transformation matrix is a 2×3 matrix, consisting of a 2×2 scaling and rotation portion and a 2×1 translation portion. Utility functions (EVALUATE TRANSFORMATION MATRIX, ACCUMULATE TRANSFORMATION MATRIX) are available to the application program for setting up the transformation matrices. A fixed point for scaling and rotation, and a shift vector in either WC or NDC may be specified. In the former case, the WC values of the fixed point and shift vector are first transformed using the current normalization transformation.

The segment transformation takes place after the normalization transformation, but before any clipping.