

© British Standards Institution. No part of this publication may be photocopied or otherwise reproduced without the prior permission in writing of BSI

Draft for Development

Computer graphics: Graphical Kernel System for three dimensions (GKS-3D) functional description

(ISO/DP 8805.2)

**Projet à développer
Infographie
GKS pour trois dimensions (GKS - 3D)
Description fonctionnelle**

**Vornorm
Graphische Datenverarbeitung;
graphisches Kernsystem für drei Dimensionen (GKS - 3D);
Funktionsbeschreibung**

Contents

National foreword	BS page
Committees responsible	5
	Back cover
 Draft for Development	
0R Introduction	6
1R Scope and field of application	9
2R References	11
3C Definitions	12
4 GKS-3D	16
4.1 About this standard	16
4.1.1R Specification	16
4.1.2 Registration	16
4.2 Introduction	17
4.3 Concepts	18
4.3.1 GKS-3D concepts	18
4.3.2 Relationship to ISO 7942 (GKS)	20
4.4 Graphical Output	21
4.4.1 Output primitives	21
4.4.2R Output primitive attributes	21
4.4.3 Polyline attributes	28
4.4.4 Polymarker attributes	28
4.4.5C Text attributes	28
4.4.6C Fill area attributes	31
4.4.7 Cell array attributes	32
4.4.8 Generalized drawing primitive attributes	32
4.4.9 Colour	32
4.4.10 Fill area set attributes	32
4.4.11 View index	33
4.4.12 Hidden line/hidden surface removal (HLHSR) identifier	33
4.5 Workstations	34
4.5.1C Workstation characteristics	34
4.5.2 Selecting a workstation	35
4.5.3R Deferring picture changes	35
4.5.4 Clearing the display surface	39
4.5.5 Elimination of primitives outside segments	39
4.5.6 Sending messages to a workstation	39
4.5.7 Hidden line/hidden surface removal	39
4.6 Coordinate Systems and Transformations	41
4.6.1C Normalization transformations	41
4.6.2R Clipping	41
4.6.3R Workstation transformations	42
4.6.4R Transformation of locator input	45
4.6.5R Transformation of stroke input	45
4.6.6 Viewing	48
4.7 Segments	57
4.7.1 Introduction to segments	57
4.7.2 Segment attributes	57
4.7.3R Segment transformations	57
4.7.4R Clipping and WDSS	57
4.7.5 Workstation Independent Segment Storage	58
4.7.6R WISS functions and clipping	58
4.8 Graphical Input	59
4.8.1R Introduction to logical input devices	59
4.8.2 Logical input device model	60

4.8.3	Operating modes of logical input devices	60
4.8.4R	Measures of each input class	60
4.8.5C	Input queue and current event report	62
4.8.6	Initialization of input devices.	62
4.8.7	Locator and stroke input using 2D input devices	63
4.9	GKS-3D Metafile Interface	65
4.10	GKS-3D Levels	66
4.10.1	Introduction	66
4.10.2C	The level structure	66
4.10.3C	Level functionality	66
4.11	States of GKS-3D and Inquiry Functions	67
4.11.1	Description of states	67
4.11.2	Inquiry functions	67
4.12	Error Handling	68
4.13	Special Interfaces	69
4.14	Interpretation of GKS functions in 3D	70
5	GKS-3D functions	74
5.1	Notational conventions	74
5.2C	Control functions	75
5.3	Output functions	76
5.4	Output Attributes for 3D	85
5.4.1	Workstation independent primitive attributes	85
5.4.2	Workstation attributes (representations)	88
5.5	Transformation Functions	90
5.5.1	Normalization transformation	90
5.5.2	Workstation transformation	91
5.6	Segment Functions	96
5.6.1	Segment manipulation functions	96
5.6.2	Segment attributes	97
5.7	Input Functions	98
5.7.1	Initialisation of input devices	98
5.7.2	Setting mode of input devices	106
5.7.3	Request input functions	106
5.7.4	Sample input functions	107
5.7.5	Event input functions	109
5.8	Metafile Functions	111
5.9	Inquiry Functions	112
5.9.1	Introduction	112
5.9.2	Inquiry functions for Operating State Value	112
5.9.3	Inquiry functions for GKS-3D Description Table	112
5.9.4	Inquiry functions for GKS-3D State List	112
5.9.5	Inquiry functions for GKS-3D workstation state list	114
5.9.6	Inquiry functions for workstation description table	124
5.9.7	Inquiry functions for segment state list	135
5.9.8	Pixel inquiries	136
5.10	Utility Functions	139
5.11	Error Handling	142
6	GKS-3D Data Structures	143
6.1	Notation and data types	143
6.2	Operating State	144
6.3	GKS-3D Description Table	145
6.4C	GKS-3D State List	146
6.5C	Workstation State List	148
6.6C	Workstation Description Table	150

6.7C	Segment State List	152
6.8	Error State List	153

Annexes

A	Function lists	154
A.1	Alphabetic	154
A.2	Order of appearance	155
A.2.1	Control functions 5.2	155
A.2.2	Output functions 5.3	155
A.2.3	Output attributes 5.4	156
A.2.4	Transformation functions 5.5	156
A.2.5	Segment functions 5.6	156
A.2.6	Input functions 5.7	157
A.2.7	Metafile functions 5.8	157
A.2.8	Inquiry functions 5.9	157
A.2.9	Utility functions 5.10	159
A.2.10	Error handling 5.11	159
A.3	Ordered by level	159
A.3.1	Level 0a	159
A.3.5	Level 0b	160
A.3.3	Level 0c	160
A.3.4	Level 1a	160
A.3.5	Level 1b	160
A.3.6	Level 2a	161
A.4	Ordered by state	161
A.4.1	Functions allowed in state GKCL	161
A.4.2	Functions allowed in state GKOP	161
A.4.3	Functions not allowed in state WSOP	161
A.4.4	Functions not allowed in state WSAC	161
A.4.5	Functions not allowed in state SGOP	161
A.5	Applicability to workstation groups	162
B	Error lists	165
B.1	Implementation dependent	165
B.2	Status	165
B.3	Workstations	165
B.4	Transformations	165
B.5	Output attributes	165
B.6	Output primitives	165
B.7	Segments	165
B.8	Input	165
B.9	Metafiles	165
B.10	Escape	165
B.11	Miscellaneous	165
B.12	System	165
B.13	Reserved errors	165
B.14	GKS-3D Transformations	165
B.15	GKS-3D output attributes	165
B.16	GKS-3D output primitives	166
C	Interfaces	167
C.1	Introduction to GKS-3D	167
C.2	Language Binding	167

D	Allowable differences in GKS-3D implementations	168
D.1	General	168
D.2C	Global differences	168
D.3C	Workstation dependent differences	168
E	Metafile structure	170
E.1	Metafiles	170
E.1.1	General	170
E.1.2	ISO 8632 Metafile	170
E.1.3	Metafile designed for GKS-3D*	170
E.2	File Format and Data Format	170
E.3C	Generation of metafiles	170
E.4	Interpretation of metafiles	174
E.4.1	General	174
E.4.2	Control items	174
E.4.3	Output primitives	174
E.4.4	Output primitive attributes	174
E.4.5	Workstation attributes	174
E.4.6	Transformations	174
E.4.7	Segment manipulation	174
E.4.8	Segment attributes	174
E.5	Control Items	174
E.6	Items for Output Primitives	174
E.7	Items for Output Primitive Attributes	176
E.8	Items for Workstation Attributes	177
E.9	Items for Transformations	177
E.10	Items for Segment Manipulation	179
E.11	Items for Segment Attributes	179
E.12	User items	179
F	Sample programs	180
G	GKS-3D functions summary	181
G.1	Control Functions	181
G.2	Output Functions	181
G.3	Output Attributes	181
G.3.1	Workstation independent primitive attributes	181
G.3.2	Workstation attributes (representations)	182
G.4	Transformation Functions	182
G.4.1	Normalization transformation	182
G.4.2	Workstation transformations	182
G.5	Segment Functions	182
G.5.1	Segment manipulation functions	183
G.5.2	Segment attributes	183
G.6	Input Functions	183
G.6.1	Initialisation of input devices	183
G.6.2	Setting the mode of input devices	183
G.6.3	Request input functions	183
G.6.4	Sample input functions	183
G.6.5	Event input functions	184
G.8C	Inquiry Functions	184
G.9	Utility Functions	184
G.10	Error Handling	184

*Numbered incorrectly in the text as E.1.2.

National foreword

This Draft for Development has been prepared under the direction of the Office and Information Standards Committee. The primary objective of this Draft for Development is to give advance guidance to product and systems designers concerned with computer graphics.

The text of the Draft for Development is identical with the International 2nd Draft Proposal ISO/DP 8805, issued by the International Organization for Standardization (ISO).

For ease of reproduction, the ISO text has been reproduced unchanged.

The UK is actively participating through BSI Technical Committee IST/21* in the development of the ISO Draft Proposal. The Technical Committee will review any agreed amendment or addenda to the ISO text and will, where appropriate, update the text of this Draft for Development via an amendment or revision. When the ISO draft reaches an appropriate level of agreement and stability, this Draft for Development may then be converted to a British Standard.

Attention is drawn to the first paragraph of the introduction which explains that the text of this Draft for Development is presented in the form of changes to BS 6390 : 1985 in order to extend the Graphical Kernel System to three dimensions, and is not usable without the text of BS 6390 : 1985.

Drafts for Development for language bindings to GKS-3D will be published when suitable texts become available. The first of these (DD 150/1 GKS-3D/Fortran) is scheduled for publication in April 1986.

A brochure giving details of existing and projected British Standards publications for Open Systems Interconnection (OSI) may be obtained from the Sales Department, BSI, Linford Wood, Milton Keynes MK14 6LE at no charge. Press releases for new OSI, data communications and computer graphics British Standards publications will be issued to the computing press, and details of all new publications will be recorded in full in BSI News.

Terminology and conventions. The text of the Draft Proposal has been approved as suitable for publication as a Draft for Development without deviation. Some terminology and certain conventions are not identical with those used in Drafts for Development; attention is drawn especially to the following.

Wherever the words 'International Standard' appear, referring to this Draft for Development, they should be read as 'Draft for Development'.

Wherever page numbers are quoted, they are ISO page numbers.

Cross-references

International standards	Corresponding British Standards
ISO 646-1983	BS 4730 : 1985 Specification for UK 7-bit data code (Identical)
ISO 7942-1985	BS 6390 : 1985 Specification for a set of functions for computer graphics programming, the Graphical Kernel System (GKS) (Identical)
ISO/DIS 8632	DD 107 Computer graphics: metafile for transfer and storage of picture description information Part 1 : 1986 Functional specification Part 2 : 1986 Character encoding Part 3 : 1986 Binary encoding Part 4 : 1986 Clear text encoding

ISO 2022 : 1982, to which reference is made in the text is currently being revised, and this revision may be implemented as a British Standard. BS 4953, which was technically equivalent to ISO 2022 : 1973, has been withdrawn (June 1984).

This publication is not to be regarded as a British Standard.

It is being issued in the Draft for Development series of publications and is of a provisional nature because the subject matter is in the progress of ongoing development within the International Standards Committee (ISO/TC 97/ SC 21 Information Retrieval, Transfer and Management for Open Systems Interconnection).

It should be applied on this provisional basis, so that information and experience of its practical application may be obtained.

A review of this Draft for Development will be carried out not later than two years after its publication. Notification of the start of the review period, with a request for the submission of comments from users of this Draft for Development will be made in an announcement in the appropriate issue of *BSI News*. Observations which it is felt should receive attention before the official call for comments will be welcomed.

According to the replies received, the responsible BSI Committee will judge whether the Draft for Development can be converted into a British Standard or what other action should be taken.

*Previously OIS/121.

0R Introduction

This extension to GKS is specified by:

- 1) replacing clauses 0, 1 and 2 of GKS (ISO 7942) by the new clauses 0, 1 and 2 contained herein; and
- 2) changing clauses 3,4,5,6 and Annexes A through G in ISO 7942. If a clause or subclause number in this document is followed by the letter 'R' (e.g. 4.1R), the text in this document replaces that in ISO 7942. If a clause or subclause number in this document is followed by the letter 'C' (e.g. 4.1C), some text as indicated in bold changes that in ISO 7942; otherwise the text in this document is an addition to the end of the corresponding clause in ISO 7942. References to the Graphical Kernel System and GKS in ISO 7942 should be read as the Graphical Kernel System for Three Dimensions and GKS-3D respectively.

GKS-3D provides a set of functions for computer graphics programming. GKS-3D is a graphics system that can be used by the majority of applications that produce computer generated pictures.

The main reasons for introducing a standard for 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-3D design was based on the following requirements:

- d) GKS-3D 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, colour displays, and 3D display devices should be controlled by GKS-3D in a uniform way.
- f) GKS-3D should provide all the capabilities required by a majority of applications.

These requirements were used to formulate a number of principles used to judge specific design alternatives. Thus it was possible to contribute to the overall design goals while focusing 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-3D should not be mutually contradictory;
 - 2) compatibility: other standards or commonly accepted rules of practice should not be violated;
 - 3) orthogonality: the functions of modules of GKS-3D, should be independent of each other, or the dependency should be structured and well

[ISO page 1]

defined.

- h) Functional capabilities: the following principles were used to define the extent of GKS-3D:
- 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-3D are organized in nine levels. An implementation of GKS-3D 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-3D should allow the design of a desirable user interface;
 - 2) clarity: the concepts and functional capabilities of GKS-3D 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-3D. 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-3D 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-3D:
- 1) device independence: GKS-3D 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-3D.

These principles led to a fundamental concept underlying the GKS-3D architecture: the concept of multiple independent graphical workstations connected to and driven

by GKS-3D. The application program can inquire the capabilities of every workstation. The GKS-3D design includes escape functions that are easily identifiable with 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-3D:
 - 1) implementability: it should be possible to support the GKS-3D 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-3D from all ISO standard programming languages.
 - 3) efficiency: GKS-3D 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-3D 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.

- l) Relationship to ISO 7942: this extension adheres to the following design goals with respect to GKS (ISO 7942):
 - 1) The relationship between the functions in ISO 7942 and the functions described in this extension should be precisely defined and not be implementation dependent;
 - 2) the format of existing functions and their parameters (as presented in clause 5 of ISO 7942) will remain unchanged in this extension;
 - 3) additional functions will be added solely to support the new 3D functionality.
- m) Compatibility of GKS Application Programs: a major goal of this extension to GKS is that application programs written using the functions specified in ISO 7942 will run without modification on implementations conforming to this extension and produce the same results on workstations of the same type, subject to the allowable differences as detailed in Annex D. However, it is recognized that in certain situations, especially involving the use of metafiles, this goal cannot always be achieved (see Annex D for further details).

1R Scope and Field of Application

This International Standard specifies a set of functions for computer graphics programming, the Graphical Kernel System for Three Dimensions (GKS-3D). GKS-3D is a basic graphics system for applications that produce computer generated pictures representing three dimensional graphical data on 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-3D 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.

This extension to GKS provides application programs with the capability to define and display 3D graphical primitives, specified using 3D coordinates. In addition, the GKS input model has been extended to provide 3D locator and stroke input.

The output primitives of this extension — the new primitives — differ from the output primitives of GKS (ISO 7942) — the old primitives — in the following ways:

- a) the polyline and polymarker primitives are defined by an arbitrary sequence of 3D coordinates;
- b) the text, fill area, and cell array primitives remain planar, but may lie in a plane positioned arbitrarily in 3D space.

New functions are provided in this extension to generate these new primitives.

The existing functions of GKS can continue to be used in the extension. However, the output functions have new meanings in that they generate instances of the new primitives — instances that are constrained to lie in a plane. Furthermore, additional capabilities are provided in this extension to:

- c) specify views of 3D objects;
- d) obtain information from 3D input devices;
- e) permit workstations to perform hidden line/hidden surface removal (HLHSR).

This extension provides no specific functions for controlling such rendering techniques as light source, shading, texturing, and shadow computations. However, workstations may apply these techniques, locally as the primitives and attributes are realized, provided that the general guidelines of Annex D concerning allowable workstation differences are followed.

For certain parameters of the functions, GKS-3D defines value ranges as being reserved for registration. The meanings of these values will be defined using the established procedures (see 4.1.2).

GKS-3D defines a language independent nucleus of a graphics system. For integration into a programming language, GKS-3D is embedded in a language dependent layer obeying the particular conventions of that language.

DD 128:1986

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

[ISO page 5]

 **Documents**
This publication is a Draft for Development and is therefore not to be regarded as a British Standard. See page 5.

2R References

- ISO 646 7-bit Coded Character Set for Information Processing Interchange
- ISO 2022 Information processing - ISO 7-bit and 8-bit Coded Character Sets - code extension techniques
- ISO 2382/13 Data processing - Vocabulary - Section 13: Computer Graphics
- ISO 6093 Information processing - Representation of Numerical Values in Character Strings for Information Interchange¹⁾
- ISO 8632 Information processing - Computer Graphics - Metafile for Transfer and Storage of Picture Description Information
- Part 1: Functional description¹⁾
- Part 2: Character encoding¹⁾
- Part 3: Binary encoding¹⁾
- Part 4: Clear text encoding¹⁾
- ISO 7942 Information Processing - Computer Graphics - Graphical Kernel System (GKS) functional description

¹⁾ At present at the stage of draft

3C Definitions

Replace the following definitions:

3.2 aspect ratio: The ratio of x to y or y to z of a volume. Used to describe the shape in a particular coordinate system (for example, of a workstation window or of a workstation viewport).

3.7 bundle table: Entries in this workstation dependent table specify workstation dependent aspects of one or more primitives. GKS-3D has polyline, polymarker, text, fill area, edge and view bundle tables.

3.9 cell array: A GKS-3D output primitive consisting of a parallelogram of equal sized cells, each of which is a parallelogram and has a single colour.

3.27 fill area bundle table: A table associating specific values for workstation dependent aspects of fill area and fill area set primitives with bundle table index values. In GKS-3D, this table contains entries for interior style, style index, and colour index. The workstation dependent aspects of the fill area set edge are contained in the edge bundle table.

3.37 locator device: A GKS-3D logical input device providing a position in world coordinates, a normalization transformation, and a viewing transformation.

3.44 normalization 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-3D this transformation maps positions in world coordinates to normalized device coordinates.

3.47 output primitive; graphic primitive; display elements: A basic graphic element that can be used to construct a display image. Output primitives in GKS-3D are polyline, polymarker, text, fill area, fill area set, cell array and generalized drawing primitive.

3.58 rotation: Turning all or part of a display image about an axis. In GKS-3D, this capability is available via the segment and viewing transformations.

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-3D, this capability is available via the segment and viewing transformations.

3.65 stroke device: A GKS-3D logical input device providing a sequence of points in world coordinates, a normalization transformation number and a viewing transformation number.

3.72 viewport: A rectangular parallelepiped volume of normalized device coordinate space.

[ISO page 7]

3.73 window: A rectangular parallelepiped volume of world coordinate space.

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 x and y but not in z. In GKS-3D, this transformation maps positions in normalized projection 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 window: A rectangular parallelepiped within normalized projection coordinates that is represented in the workstation display space.

Add the following definitions:

back clipping plane: A plane parallel to the view plane whose location is specified by a viewing reference coordinate distance along the view plane normal from the view reference point. When back plane clipping is enabled, portions of objects behind the back clipping plane are discarded. Clipping against the back clipping plane is controlled separately from all other clipping.

back plane clipping: Clipping against the back clipping plane. If back plane clipping is enabled, only that part of the object that lies in front of and on the back clipping plane will be projected onto the viewport.

centre of projection: See Projection Reference Point (PRP).

clipping volume: Either the normalization viewport clipping volume, the view clipping volume or the intersection of the both. The normalization viewport clipping volume is a rectangular parallelepiped in normalized device coordinates (NDC3). The view clipping volume is in the viewing reference coordinate system and is defined by the window, front and back clipping planes, and projectors of the corners of the window. Graphics data within these volume will not be clipped. Data on the planes forming the edges of these volumes are considered to be within the volume.

edge: The boundaries of all polygons of a fill area set primitive. The appearance of an edge is controlled by a set of edge aspects and the workstation edge representation.

edge bundle table: A table associating specific values for all workstation dependent aspects of an edge bundle index. In GKS-3D, this table contains entries consisting of edge flag, edgetype, edge width scale factor, and colour index.

fill area set: A GKS-3D output primitive consisting of a set of polygons which may be hollow or may be filled with a uniform colour, a pattern, or a hatch style. Edge control is provided.

front clipping plane: A plane parallel to the view plane that is specified by a distance in viewing reference coordinates along the view plane normal from the view reference point. When front plane clipping is enabled, portions of objects in front of the front clipping plane are discarded. Clipping against the front clipping plane is controlled separately from window clipping.

front plane clipping: Front plane clipping is performed against the part of the object that lies within and on the view volume, behind and on the front clipping plane. Front plane clipping is independent of back plane clipping.

leading edge: For text path LEFT and RIGHT, either the left or right side of the character body, depending on which side is closest to the text position. If both sides are equidistant, the leading edge is the left side of the character body. For text path UP and DOWN, either the top or bottom of the character body, again depending on which side is closest to the text position. If both sides are equidistant, the leading edge is the top of the character body.

oblique view: A parallel projection whose projectors are not perpendicular to the view plane.

parallel projection: The image of a three dimensional object in which lines that are parallel in the object appear parallel in the image without regard to relative distance or depth.

perspective projection: The image of a three dimensional object in which parallel lines appear to converge in the image as a function of relative distance or depth of the object from the position of the viewer.

projector: Lines that pass through each point of an output primitive and intersect the view plane.

projection plane: See view plane.

projection reference point: A point in viewing reference coordinates (VRC) that determines the direction of all projectors in a parallel projection (i.e., all projectors are parallel to the vector defined by the projection reference point and the window center) or from which all projectors in a perspective projection emanate.

view index: The attribute of all primitives that selects the viewing representation applied to the primitives at display time.

viewing coordinate system; view reference coordinate system A three dimensional coordinate system used for defining windows and projection parameters. The origin of the viewing coordinate system is the intersection of a line through the view reference point parallel to the view plane normal vector with the view plane. The axes of the viewing coordinate system are labeled U, V, and N. The N coordinate axis is parallel to the view plane normal vector. The positive V axis is defined to be the projection of the view up vector onto the view plane. The U coordinate axis is selected to define an orthogonal right-handed coordinate system. Also called the UVN Coordinate System.

viewing transformation: A transformation that maps positions in NDC3 coordinates to positions in view reference coordinates.

view matrix: A 4×4 matrix used to specify the viewing transformation. The view matrix may be composed using utility functions.

view plane: A two dimensional plane through which three dimensional objects are projected. The view plane is established by giving a view reference point, a view plane normal, and a view plane distance.

view plane distance: The distance of the view plane from the view reference point measured along the view plane normal vector in view reference system coordinates.

view plane normal: A vector normal to the view plane used to orient the plane.

view reference point: A convenient world coordinate point on or near the object being viewed. All viewing parameters are specified relative to the view reference point. The view reference point becomes the origin of the viewing coordinate system.

view table: The workstation dependent table of views found on each workstation. Referenced by the workstation independent view index attribute.

view up vector: A vector in world coordinates relative to the view reference point that, if it were within the window, would appear upright on the display surface. The view up vector is projected onto the view plane, in the direction of the view plane normal; this projection (which is the V axis in the viewing coordinate system) orients the world coordinate "up" direction for the view.

view volume: The view volume is either a frustrum for perspective projection or a parallelepiped for parallel projection. The boundaries of the view volume are determined by the window defined in the view plane, the front and back clipping planes, and projectors from the projection reference point through the window corners (for perspective projection) or projectors in the direction of projection through the window corners (for parallel projection). If the projection reference point is not on the view plane normal, a sheared projection will result. In strictly two dimensional applications, the view volume reduces to a window on the view plane.

view window: The boundaries of a rectangle on the view plane which partially defines the view volume. Also used to refer to the entire view volume.

Renumber the definitions as 3.1 through 3.109.

4 GKS-3D

4.1 About this standard

4.1.1R Specification

The set of functions known as the Graphical Kernel System for Three Dimensions (GKS-3D) 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. The meaning of the GKS functions described in ISO 7942 are changed to apply to a three dimensional environment. An implementation of GKS-3D shall contain precisely the functions of one level. A GKS-3D 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-3D functions shall be used only via GKS-3D.

4.1.2 Registration

[ISO page 11]

This publication is a Draft for Development and is therefore not to be regarded as a British Standard. See page 5b

4.2 Introduction

In addition, GKS-3D provides a fill area set primitive.

4.3R Concepts

4.3.1 GKS-3D concepts

The graphical output that is generated by GKS-3D 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 line style, 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-3D 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 prompt and echoes, is controlled by GKS-3D for each logical input device individually.

The two abstract concepts (abstract output and abstract input) are the building blocks of an *abstract workstation*. A GKS-3D workstation represents a unit consisting of zero or one display spaces 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. Conceptually, all workstations are three dimensional in GKS-3D.

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

- a) World Coordinates (WC3), used by the application programmer;
- b) Normalized Device Coordinates (NDC3), used to define a uniform coordinate system for all workstations;
- c) View Reference Coordinates (VRC), used to define the parameters of a view;
- d) Normalized Projection Coordinates (NPC), used to facilitate hidden line/hidden surface computations and composition of different views;
- e) Device Coordinates (DC3), one coordinate system per workstation, representing its display space.

Output primitives and attributes are mapped from WC3 to NDC3 by normalization transformations, from NDC3 to NDC3 by segment transformations (see next paragraph), from NDC3 to VRC by the view transformation, from VRC to NPC by the projection transformation and finally from NPC to DC3 by the workstation transformation. Locator and stroke input is mapped by the inverse of these transformations from DC3 to WC3.

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 space 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 and fill area set. In addition, the non-geometric aspects of primitives are controlled by the primitive attributes in one of two ways. A non-geometric aspect of a primitive can be specified either by an attribute that specifies one or more aspects of the primitive by an index which points to a workstation dependent representation (set of values) or by an attribute that specifies each aspect independent of the workstation. The former is referred to as bundled specification and the latter is referred to as individual specification. In addition, an index is bound to the primitive which selects the view and projection transformations in a workstation dependent manner.

Hidden lines or hidden surfaces may be removed using a workstation dependent method (see 4.5.7).

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 linetype, 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 appearance of segments is controlled by the segment attributes. The segment attributes 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 that control the operation of logical input devices can be specified either when a device is initialised or as part of the input device setting. Through initialisation, an initial value, a prompt and echo technique, and an area on the device for echoing can be specified. A data record may provide further device-specific input 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-3D can be in one of five operating states. Associated with each state are the set of GKS-3D 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 for 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-3D provides an interface to a system for filing graphical information for external long term storage and exchange. The interface consists of a GKS-3D Metafile output workstation, which writes to a so-called graphics metafile (which is sequential), and a GKS-3D Metafile input workstation, which reads from the metafile. In addition to the normal functions for output to workstations, a GKS-3D 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.

4.3.2 Relationship to ISO 7942 (GKS)

GKS-3D provides graphic functions for primitives defined in a three dimensional space whereas GKS provides functions to primitives defined in a plane. Thus, GKS functions are a special case of the corresponding functions in GKS-3D.

GKS-3D has a set of output primitives (see 4.4.1) which can be generated either by a GKS-3D function or by a GKS function (e.g. a polyline primitive is generated by the GKS-3D function POLYLINE 3 or the GKS function POLYLINE). However, GKS functions used in a GKS-3D environment can be affected by GKS-3D functions, for example, primitives generated using GKS functions will be transformed by 3D transformation matrices.

The primitives generated by the GKS functions are considered to lie in the $Z=0$ plane. The default view is an orthographic projection onto the $Z=0$ plane. The implication is that GKS programs running under GKS-3D can produce the same results.

The mapping between corresponding functions of GKS and GKS-3D is described in 4.14.

4.4 Graphical Output

4.4.1R Output primitives

The graphical information that is generated by GKS-3D and routed to all active workstations is composed of basic pieces called output primitives. GKS-3D provides seven output primitives:

- a) POLYLINE: GKS-3D generates a set of connected lines defined by a point sequence.
- b) POLYMARKER: GKS-3D generates symbols of one type centred at given positions.
- c) TEXT: GKS-3D generates a character string at a given position.
- d) FILL AREA: GKS-3D generates a polygonal area, which may be hollow or filled with a uniform colour, pattern, or hatch style.
- e) FILL AREA SET: GKS-3D generates a set of polygonal areas which may be hollow or filled with a uniform colour, pattern, or hatch style. This allows for specifying areas with holes or disjoint regions that must be treated as a single entity, particularly for hidden line/hidden surface removal computation.
- f) CELL ARRAY: GKS-3D generates an array of cells with individual colours.
- g) GENERALIZED DRAWING PRIMITIVE (GDP):
GKS-3D addresses special geometrical output capabilities of a workstation such as the drawing of spline curves, circular arcs, and elliptical arcs. The objects are characterized by an identifier, a set of points and additional data. GKS-3D applies all transformations to the points but leaves the interpretation to the workstation.

Text, fill area, fill area set, and cell array are planar primitives in arbitrary planes. Conceptually these planes have 2 sides and zero thickness. The viewing transformation determines which side of the primitive is seen.

The plane in which the text primitive lies is defined by a point and two vectors that are contained in the definition of the text primitive. Similarly, the plane in which the cell array primitive lies is defined by three points contained in the definition of the cell array primitive. For fill area and fill area set primitives, the plane of the primitives is defined by the points of the respective primitives.

Insuring that the points are coplanar is the responsibility of the application program. Implementations are not required to check coplanarity. While slightly non-coplanar points (e.g. owing to numerical roundoff errors) should be mapped to a plane, the effect of large divergences is implementation dependent.

4.4.2R Output primitive attributes

Each output primitive potentially has four types of attributes (geometric, non-geometric, viewing and identification). The first two attribute types determine the exact appearance of the output primitive while the fourth attribute type is used with

input. The third attribute type selects the viewing parameters for the workstation dependent sections of the pipeline.

The values of these attributes are set modally and are recorded in the GKS-3D state list. A separate function is provided for each primitive attribute (except the ASFs: see later in this subclause) 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 an output primitive function 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 that affect the shape or size of the whole primitive (for example, character height for text). Hence, they are sometimes referred to as geometric attributes. Geometric attributes are workstation independent and, if they represent coordinate data (points or displacements), these 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 several geometric attributes.

Current values of (workstation independent) geometric attributes are expressed in world coordinates and 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, viewing and workstation transformation.

Attributes of the second type control the non-geometric aspects of primitives. These are aspects that 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 poly-marker). Non-geometric aspects do not represent coordinate data. The non-geometric aspects of a primitive may be specified either via a bundle or individually.

Bundled aspects are controlled by a bundle index into a bundle table, each entry of which contains non-geometric aspects of a primitive. 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-3D state list.

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. Because the bundles are set separately for each workstation, the values of their components are restricted to the valid values for that workstation. For 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 described 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 most similar to 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.

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-3D state list. There is one ASF for each non-geometric aspect. 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-3D is open by the function **SET ASPECT SOURCE FLAGS** or **SET ASPECT SOURCE FLAGS 3**. 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 the primitive.
- d) If the ASF for an aspect is **BUNDLED**, the value used on a workstation is the value of the corresponding component of the workstation bundle, pointed to by the bundle index,

Colour is a non-geometric aspect of a primitive, which 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, linetypes for polyline).

There are two attributes of the third type, namely view index and HLHSR identifier. The attribute view index is an entry in a view bundle table at a workstation. There is one view bundle table per workstation.

The attribute HLHSR identifier supplies HLHSR information to the workstation. The interpretation of the identifier is implementation dependent (see 4.5.7). Implementation of the functions to control hidden line/hidden surface removal is obligatory,

but the functions may have no effect on the appearance of primitives.

There is precisely one attribute of the fourth 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 that apply to each output primitive (attributes controlling non-geometric aspects, geometric attributes, view 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
VIEW INDEX
HLHSR IDENTIFIER
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
VIEW INDEX
HLHSR IDENTIFIER
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
VIEW INDEX
HLHSR IDENTIFIER
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 POINTS |
| | VIEW INDEX |
| | HLHSR IDENTIFIER |
| | PICK IDENTIFIER |
| i) FILL AREA SET: | 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 POINTS |
| | EDGE INDEX |
| | EDGE FLAG |
| | EDGETYPE |
| | EDGEWIDTH SCALE FACTOR |
| | EDGE COLOUR INDEX |
| | EDGE FLAG ASF |
| | EDGETYPE ASF |
| | EDGEWIDTH SCALE FACTOR ASF |
| | EDGE COLOUR INDEX ASF |
| | VIEW INDEX |
| | HLHSR IDENTIFIER |
| | PICK IDENTIFIER |
| j) CELL ARRAY | VIEW INDEX |
| | HLHSR IDENTIFIER |
| | PICK IDENTIFIER |
| k) GENERALIZED DRAW-
ING PRIMITIVE: | Zero or more of the sets e) to i)
except that PICK IDENTIFIER,
VIEW INDEX, and HLHSR IDENTIFIER are
always attributes |

Figure 2R shows the binding of the attributes.

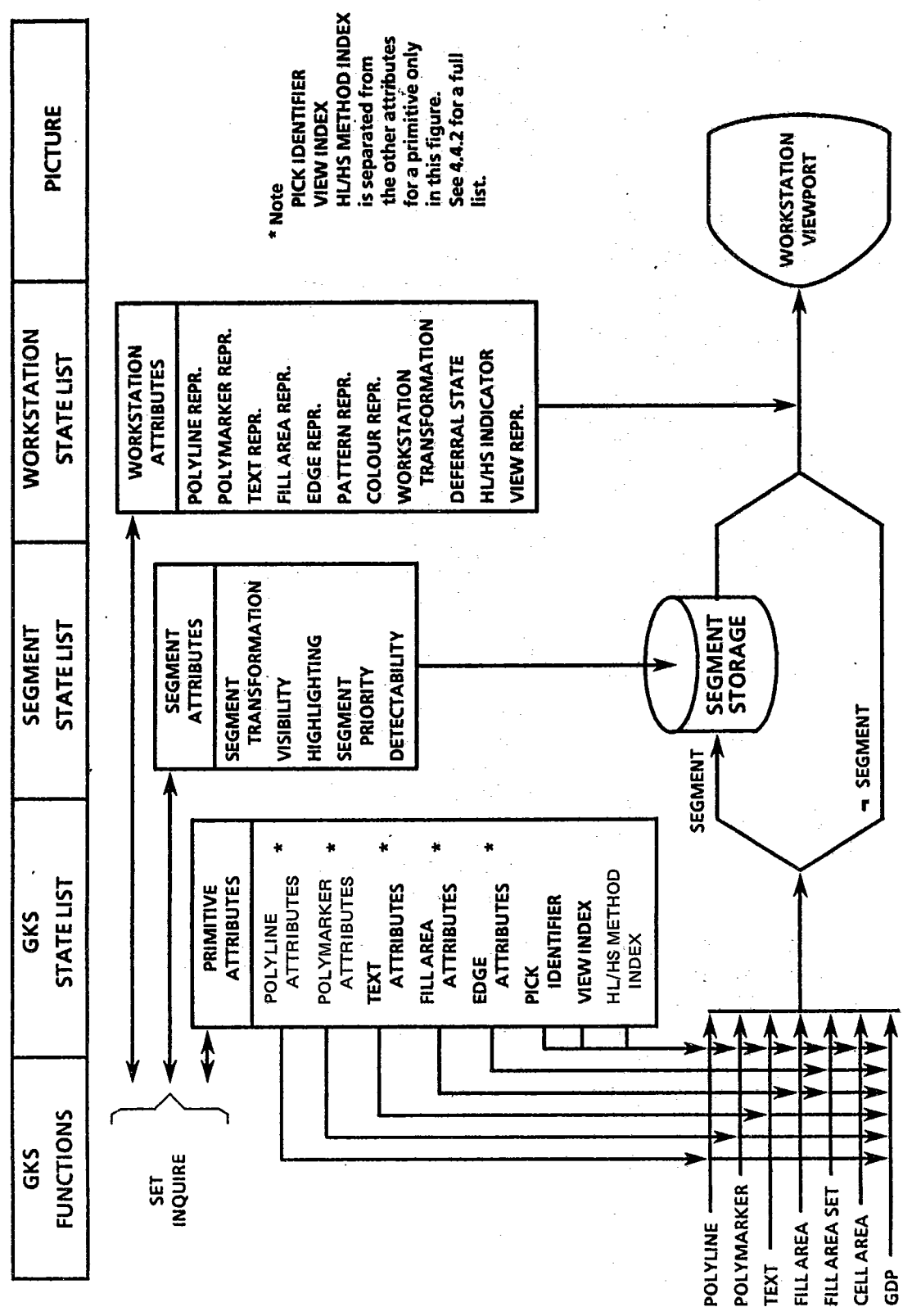


Figure 2R. Binding of attributes

The attributes for each primitive, other than PICK IDENTIFIER, are described in 4.4.3 to 4.4.8 and 4.4.10 to 4.4.12. PICK IDENTIFIER is described in more detail in 4.7.1. VIEW INDEX and HLHSR IDENTIFIER are described in more detail in sub-clauses 4.4.11 and 4.4.12 respectively. 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 <bundle> INDEX or by individually specified attributes. Non-geometric aspects appear in lower case unless the corresponding individually specified attributes are being used. Individually specified attributes appear in upper case (for example, the aspect linetype but the individually specified LINETYPE).

The entries in the bundle, pattern and colour tables may be set separately for each workstation. Some predefined 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 <bundle | VIEW | 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
- edge bundle table
- pattern table
- colour table
- view table

The values in these tables may be changed dynamically. The only way of changing the aspects of a primitive that are stored in a bundle table is by changing the bundle table entry. However, note that a change in a bundle table entry can be reflected in a displayed primitive only if the values of the corresponding primitive's ASFs for the aspects in the bundle table are BUNDLED. The entry 'dynamic modification accepted' in the workstation description table indicates which changes:

- l) lead to an implicit regeneration (may be deferred) (IRG);
- m) 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

4.4.4 Polymarker attributes

Since the polymarker has no geometric attributes, transformations affect only the marker position and not the appearance of the marker glyph.

4.4.5C Text attributes

Insert before the first paragraph:

Text is a planar primitive. The plane of the text primitive is defined by a point and two vectors contained in the definition of the text primitive. The point is called the text position. The two vectors are called the first and second text direction vectors. The point and these vectors define a text plane and a local coordinate system on that plane in which all geometric text attributes are measured.

Let P be the text position and U and V be the first and second text direction vectors respectively. The text plane is the plane that contains P and is perpendicular to $U \times V$. The origin of the local coordinate system is at the point P . The positive X axis of the local coordinate system is parallel to U . Let T be a vector perpendicular to U and in the text plane. If $T \cdot V$ is positive, then the positive Y axis is parallel to T ; otherwise the positive Y axis is opposite T . In the following all references to coordinates refer to this local coordinate system unless explicitly stated otherwise.

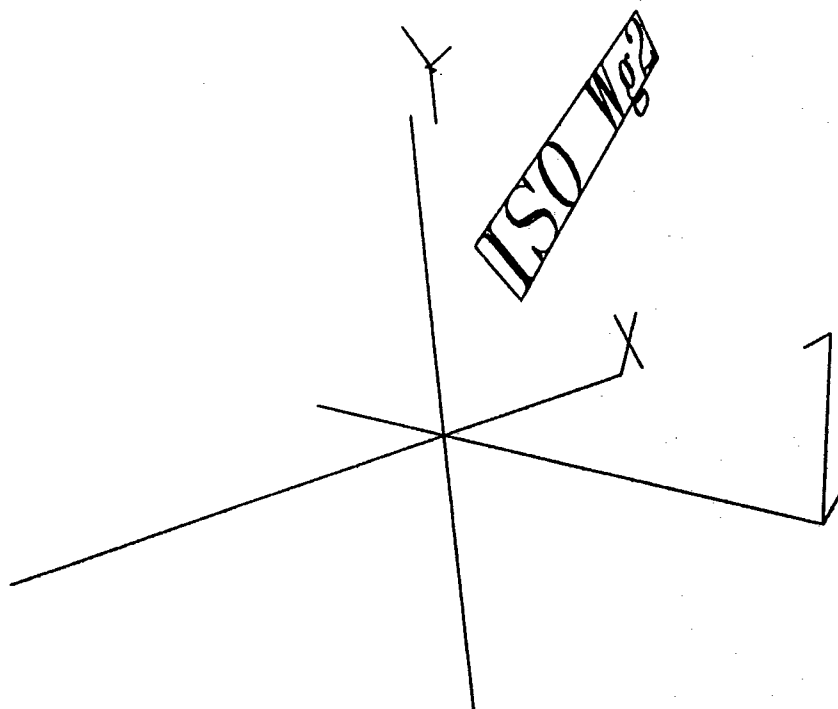
replace 17th paragraph by:

The effect to be achieved is now described. A series of rectangles, aligned and positioned on the text plane by applying the current text attributes to the nominal character body extent rectangle, is passed through the viewing pipeline. The rectangle height is CHARACTER HEIGHT and is parallel to the CHARACTER UP VECTOR. The rectangle width is CHARACTER WIDTH and is parallel to the CHARACTER WIDTH VECTOR. The world coordinates of the four corners of the rectangle are transformed by the normalization transformation, by a segment transformation if within a segment, by a viewing transformation, and by the workstation transformation. The resulting quadrilateral can be used by the workstation character generator. Thus, the shape of individual characters can be transformed by a normalization transformation that scales x , y and z non-uniformly, by a segment transformation, by a viewing transformation and by a workstation transformation that scales z non-uniformly. Before the viewing transformation, the character body extent is a parallelogram and can be conveniently represented by the text position and two vectors to the two adjacent corners of the parallelogram. This parallelogram is stored in segments.

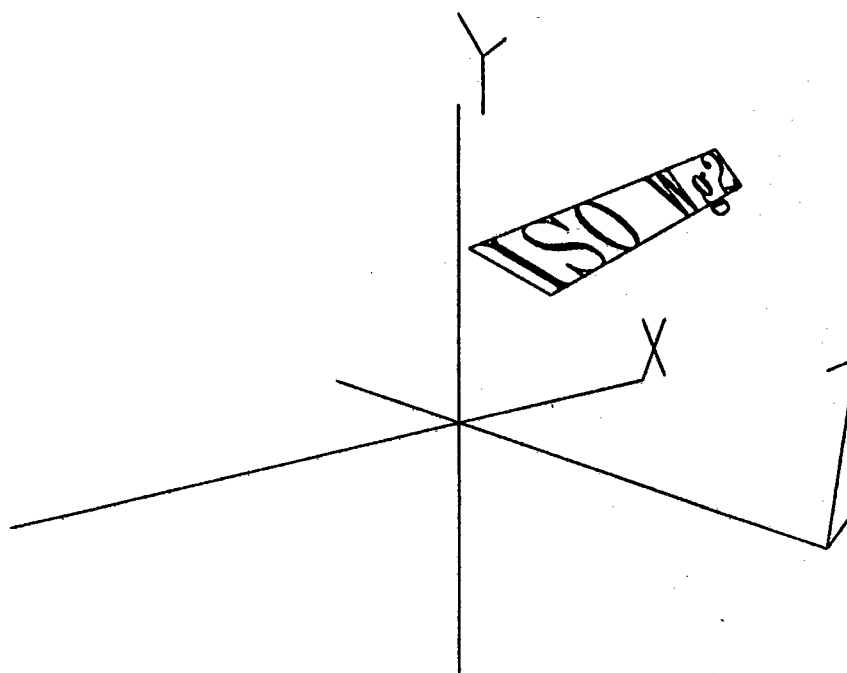
The resulting characters need conform to the shape of the character extent quadrilateral exactly for STROKE precision. For STRING precision text, the character height and width vectors need be evaluated only at the leading edge of the first character. For CHAR precision, the character height and width vectors need be evaluated only at the leading edge of each character.

add to the end of the last clause:

Figure 16 shows the effects of different normalization transformations on text in STROKE precision.



world coordinates: character up vector = $(-1, 1)$



After a normalization transformation which scales
x and z by a factor of 2 compared with y.

Figure 16. Effects of different normalization transformations on text in **STROKE** precision.

4.4.6C Fill area attributes

Replace first paragraph by:

FILL AREA has geometric attributes PATTERN REFERENCE POINTS 3 and PATTERN SIZE. It also has two implicitly specified geometric attributes PATTERN WIDTH VECTOR and PATTERN HEIGHT VECTOR. These are implicitly specified by the other geometric attributes. Like ordinary geometric attributes, their values are bound to the fill area primitives when the primitives are created and cannot be changed afterwards. 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 subclause.

Replace 4th paragraph by:

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 starting position of the pattern are determined by a pattern box in a local coordinate system on the fill area plane. The pattern box is defined as follows:

1. The three pattern reference points are projected along a normal vector onto the fill area plane.
2. The origin of the box, which may be a parallelogram, is the projection of the first point.
3. The PATTERN WIDTH VECTOR 3, which is one side of the pattern box, is the vector from the first projected pattern reference point in the direction of the second projected pattern reference point, with length equal to the SX component of the pattern width vector from the GKS-3D state list.
4. The PATTERN HEIGHT VECTOR 3, which is an adjacent side of the pattern box, is the vector from the first projected pattern reference point in the direction of the third projected pattern reference point, with length equal to the SY component of the pattern height vector from the GKS-3D state list.

The pattern box is conceptually divided into a grid of DX×DY cells. The colour index array is associated with the cells as follows: the element (1,DY) is associated with the cell having the first projected 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 pattern is mapped onto the fill area by replicating the pattern box in both X and Y directions until the entire fill area is covered. The resulting pattern is subjected to the same transformations as the fill area primitive.

If the projected pattern reference points form a degenerate case, an implementation dependent technique determines the location and shape of the pattern box.

A fill area primitive with interior style HOLLOW is transparent for HLHSR purposes.

4.4.7 Cell array attributes

The plane of the cell array primitive is defined by 3 points which are the corners of a parallelogram (See figure 23). The results are implementation dependent if the points are co-linear.

4.4.8 Generalized drawing primitive attributes

4.4.9 Colour

4.4.10 Fill area set attributes

Fill area set uses the same fill area attributes as fill area (see section 4.4.6).

Another set of aspects for fill area set permits independent control of the edges of fill area sets. The aspects, which can be either individual or bundled, are:

EDGE FLAG
EDGE TYPE
EDGE WIDTH SCALE FACTOR
EDGE COLOUR INDEX

The EDGE FLAG can have values ON or OFF. When the value is OFF, the edge is not drawn. When the value is ON, the edge is rendered according to the other EDGE attributes. Every fill area set primitive has an EDGE INDEX attribute, which selects the bundle from the workstation edge bundle table. For each edge aspect an aspect source flag exists that controls whether the individual aspect is bound to the fill area set or whether the value is taken from the bundle designated by the EDGE INDEX, as follows:

EDGE FLAG ASF
EDGE TYPE ASF
EDGE WIDTH SCALE FACTOR ASF
EDGE COLOUR INDEX ASF

The edges drawn according to the above aspects are drawn on top of the interior. Conceptually edges and interior are disjoint, in practice they may overlap. The boundary drawn for style HOLLOW is considered part of the interior. Consequently, the edges will have precedence over the interior when drawn, and will then be visible. The boundary associated with the style HOLLOW may be visible as well. Parts of edges that are clipped become invisible. Parts of HOLLOW interiors that become clipped will have a boundary drawn at the clipping boundary. A fill area set primitive with interior style HOLLOW is transparent for HLHSR purposes.

4.4.11 View index

An attribute of each primitive is the VIEW INDEX. The current value of the VIEW INDEX is bound to the primitive at creation. The index will select viewing parameters from a view bundle table at each workstation. The VIEW INDEX is treated in the same fashion as the other bundle indices. The VIEW REPRESENTATION is workstation dependent. The INSERT SEGMENT and INSERT SEGMENT 3 functions replace the VIEW INDEX, stored with an output primitive when the segment was created, by the current VIEW INDEX.

4.4.12 Hidden line/hidden surface removal (HLHSR) identifier

An attribute for each primitive is the HLHSR IDENTIFIER. The current value of the HLHSR IDENTIFIER is bound to the primitive at creation. The HLHSR IDENTIFIER will be interpreted at the workstation in an implementation dependent manner.

The INSERT SEGMENT and INSERT SEGMENT 3 functions replace the HLHSR IDENTIFIER, stored with an output primitive when the segment was created, by the current HLHSR IDENTIFIER.

4.5 Workstations

4.5.1C Workstation characteristics

Replace 3rd paragraph by:

An abstract graphical workstation with maximum capabilities:

- a) has one addressable display space of fixed size;
- b) allows only rectangular display spaces (the display space does not consist of several 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 that is required;
- f) permits REQUEST, SAMPLE and EVENT type input;
- g) allows logical input devices to be placed in REQUEST, SAMPLE and EVENT mode independently of each other;
- h) stores segments and provides facilities for changing and manipulating them;
- i) permits definition of several transformations that support 3D viewing transformations.

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

Replace 6th paragraph by:

GKS-3D allows the appearance of output primitives to vary between workstations, thus allowing advantage to be taken of their differing capabilities. The facilities that 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)
- edge representation (see 4.4)
- pattern representation (see 4.4)
- colour representation (see 4.4)
- deferral state (see 4.5.3)
- view representation (see 4.6.6)
- workstation transformation (see 4.6.3)
- HLHSR mode (see 4.5.7)

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

4.5.2 *Selecting a workstation*

4.5.3R *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. To use the capabilities of a workstation efficiently, however, GKS-3D 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 a deferral state that 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 of, not just an addition to, the picture.

The concept of deferral refers only to visible effects of GKS-3D 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-3D 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-3D, 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
POLYLINE 3
POLYMARKER
POLYMARKER 3
TEXT

TEXT 3
 FILL AREA
 FILL AREA 3
 FILL AREA SET
 FILL AREA SET 3
 CELL ARRAY
 CELL ARRAY 3
 GENERALIZED DRAWING PRIMITIVE
 GENERALIZED DRAWING PRIMITIVE 3
 INSERT SEGMENT 3
 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 space.

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 EDGE REPRESENTATION
 SET PATTERN REPRESENTATION
 SET COLOUR REPRESENTATION

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

SET VIEW REPRESENTATION
 SET WORKSTATION WINDOW
 SET WORKSTATION WINDOW 3
 SET WORKSTATION VIEWPORT
 SET WORKSTATION VIEWPORT 3

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

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

POLYLINE
 POLYLINE 3
 POLYMARKER
 POLYMARKER 3
 TEXT
 TEXT 3
 FILL AREA
 FILL AREA 3
 FILL AREA SET
 FILL AREA SET 3
 CELL ARRAY
 CELL ARRAY 3
 GENERALIZED DRAWING PRIMITIVE
 GENERALIZED DRAWING PRIMITIVE 3
 INSERT SEGMENT
 INSERT SEGMENT 3

(Because only segments have priority, adding primitives outside segments does not require an implicit regeneration.)

ii) 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 SEGMENT TRANSFORMATION 3
 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

DD 128:1986

SET SEGMENT TRANSFORMATION 3

- 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

- 9) if the 'dynamic modification accepted' entry in the workstation description table is IRG for the HLHSR MODE:

SET HLHSR MODE

- 10) if HLHSR MODE is such that the workstation will preform HLHSR and the operating state is SGOP:

POLYLINE
POLYLINE 3
POLYMARKER
POLYMARKER 3
TEXT
TEXT 3
FILL AREA
FILL AREA 3
FILL AREA SET
FILL AREA SET 3
GENERALIZED DRAWING PRIMITIVE
GENERALIZED DRAWING PRIMITIVE 3
INSERT SEGMENT 3

- 11) if HLHSR is being performed:

SET HLHSR IDENTIFIER

- 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 a listed function causes a visible effect on the display; for example, if an invisible segment is deleted, a regeneration need not be done. However, an

[ISO page 33]

This publication is a Draft for Development and is therefore not to be regarded as a British Standard. See page 5.

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

The second capability means that the display surface is cleared only when needed: this would normally be when the display surface is not clear (i.e., when the 'display space surface' entry in the workstation state list is NOTEMPTY). The entry 'display space 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 space (for example, a GDP primitive that 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

4.5.6 Sending messages to a workstation

4.5.7 Hidden line/hidden surface removal

Hidden line/hidden surface removal (HLHSR) is the process of determining which parts of primitives are invisible as a result of being obscured by other primitives when observed from the PRP or along the projection direction (see 4.6.6). In GKS-3D, the method by which these parts are calculated is implementation dependent. Moreover, implementation of any HLHSR capabilities is not mandatory.

HLHSR is controlled on a workstation by workstation basis, by the workstation attribute HLHSR MODE and, on a primitive by primitive basis, by a global attribute HLHSR IDENTIFIER. Both attributes are specified as integers the meaning of which are determined in a workstation and implementation dependent manner. The attributes may not be used for other purposes than HLHSR. The default value for both attributes is 0. Where appropriate, 0 should be the most convenient action for that workstation.

An implementation can use the attribute in several ways. For example, an implementation might allow several HLHSR methods to be used on a workstation simultaneously. The HLHSR identifier would control which of the available methods in a

predefined table would be used to process a primitive. HLHSR MODE could be used to switch between different predefined HLHSR tables and allow dynamic change between a cheap and expensive HLHSR method. A HLHSR MODE of 0 could disable HLHSR.

Another example might be when there is only one HLHSR method on a workstation. The HLHSR MODE would be used to indicate whether any processing was desired at this time. The HLHSR IDENTIFIER could be used to group the data in a way useful for the available method. For example, an implementation could use HLHSR IDENTIFIER to group the data into sets which are mapped into the same workstation window and therefore likely to be overlapped.

Adding a primitive may cause an implicit regeneration if HLHSR is being performed. Primitives outside segments do not take part in HLHSR calculations. Primitives in segments that have the segment attribute visibility set to INVISIBLE cannot obscure other primitives. Primitives that are not visible as a result of HLHSR are not detectable.

The INSERT SEGMENT and INSERT SEGMENT 3 functions replace the HLHSR IDENTIFIER bound to a primitive by the current HLHSR IDENTIFIER.

HLHSR removal takes precedence over segment priority.

4.6 Coordinate Systems and Transformations

4.6.1C Normalization transformations

Replace first paragraph by:

In GKS-3D, the application programmer can compose his graphical picture from separate parts each of which conceptually is defined within its own world coordinate system (WC3). The relative positioning of the separate parts is defined by having a single normalized device coordinate space (NDC3) onto which all the defined world coordinate systems are mapped. A set of normalization transformations define the mapping from the world coordinate systems onto the single normalized device coordinate space, which can be regarded as a workstation independent abstract viewing volume. This normalized picture can be stored and manipulated via the segment mechanism; it can also be stored on a metafile.

Replace 3rd paragraph by:

A normalization transformation is specified by defining the limits of a volume in the world coordinate system (window) that is to be mapped onto a specified volume of the normalized device coordinate space (viewport). Window and viewport limits specify rectangular parallelepipeds parallel to the coordinate axes in WC3 and NDC3. The rectangular parallelepipeds include their boundaries. The normalization transformation does a mapping from WC3 onto NDC3 that includes translation and differential scaling with positive scale factors for the three axes.

Replace 4th paragraph by:

Although NDC3 space conceptually extends to infinity, the part of NDC3 space in which the viewport must be located is the closed range $[0,1] \times [0,1] \times [0,1]$. In addition, an implementation may support only a restricted range of NDC3 space. However, this range is always sufficiently greater than the $[0,1] \times [0,1] \times [0,1]$ cube so that useful effects of INSERT SEGMENT and INSERT SEGMENT 3 can be achieved. In particular, NDC3 coordinates in the range $[-7,7] \times [-7,7] \times [-7,7]$ are always handled.

Replace 5th paragraph by:

Each normalization transformation is identified by a transformation number that is an integer between 0 and an implementation dependent value which can be inquired from the GKS-3D state list. The normalization transformation with transformation number 0 is the transformation that maps $[0,1] \times [0,1] \times [0,1]$ in WC3 to $[0,1] \times [0,1] \times [0,1]$ in NDC3. It cannot be changed.

4.6.2R Clipping

The viewport of a particular normalization transformation defines a clipping volume, as well as specifying the normalization transformation with the window. When the viewport of the current normalization transformation is set or when a normalization transformation is selected, the clipping volume entry in the GKS-3D state list is set to the resulting viewport of the current normalization transformation. Clipping to

this clipping volume can either be enabled or disabled. There is a single global switch (the clipping indicator) which defines whether the clipping volume is to be used for clipping.

It is anticipated that most applications will disable this clipping and use the view clipping mechanism instead (see 4.6.6).

Clipping does not take place when the normalization transformation is done but is delayed until the output primitives are to be displayed on the display space of a workstation. Output primitives stored in segments have their coordinates transformed to NDC3 and the associated clipping volume and clipping indicator is stored with the primitives. The INSERT SEGMENT and INSERT SEGMENT 3 functions allow the clipping volume and clipping indicator in the GKS-3D state list to replace the clipping volume stored with an output primitive when the segment was defined.

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

4.6.3R Workstation transformations

NDC3 coordinates are transformed into NPC coordinates by the viewing operations as described in 4.6.6. The NPC space can be regarded as a workstation dependent abstract viewing space. Each open workstation can select some part of its NPC space to be displayed somewhere on the workstation's physical display space. A particular workstation transformation is a mapping from NPC space into the device coordinate space (DC3) for that workstation.

The units of device coordinates (DC3) 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 DC3 origin is at the bottom left corner furthest from the observer of the display space;
- b) the device coordinate units are related to the display space in such a way that a cube in device coordinates appears as a cube in the display space (this is trivially true if device coordinate units are meters);
- c) x, y, and z increase to the right, upwards and towards the observer 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 coordinates is recorded in the workstation description table.

A workstation transformation is specified by defining the limits of a volume within NPC space within the range $[0,1] \times [0,1] \times [0,1]$ (the workstation window), which is to be mapped onto a specified volume in DC3 space (the workstation viewport). Workstation window volume and workstation viewport volume limits specify rectangular boxes

parallel to the coordinate axes in NPC and DC3. The rectangular boxes include their boundaries. To ensure that no output outside the workstation window is displayed, output is clipped at the workstation window boundaries, and this clipping cannot be disabled.

All workstation transformations are set by default to map the default NPC $[0,1] \times [0,1] \times [0,1]$ onto the maximum workstation display space.

Workstation transformations can be changed by SET WORKSTATION WINDOW, SET WORKSTATION WINDOW 3, SET WORKSTATION VIEWPORT, and SET WORKSTATION VIEWPORT 3. Since a specification of the workstation transformation may be deferred (see 4.5.3), these 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.

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

As the workstation window is defined within the NPC range $[0,1] \times [0,1] \times [0,1]$, this ensures that the only part of NPC space that can be viewed on any workstation lies in the range $[0,1] \times [0,1] \times [0,1]$. If the workstation window and workstation viewport have different aspect ratios for x, y, and z, the scaling specified would be different on each axis if the workstation window was mapped onto the workstation viewport in its entirety. To ensure optimum results on the display surface, the workstation transformation maps the workstation window onto the largest parallelepiped that can fit within the workstation viewport such that:

- a) aspect ratio in x and y is preserved;
- b) the lower left hand corner of the workstation window closest to 0.0 in z is mapped to the lower left hand corner of the workstation viewport furthest from the observer;
- c) the z extent of the workstation window is mapped to the entire z extent of the workstation viewport.

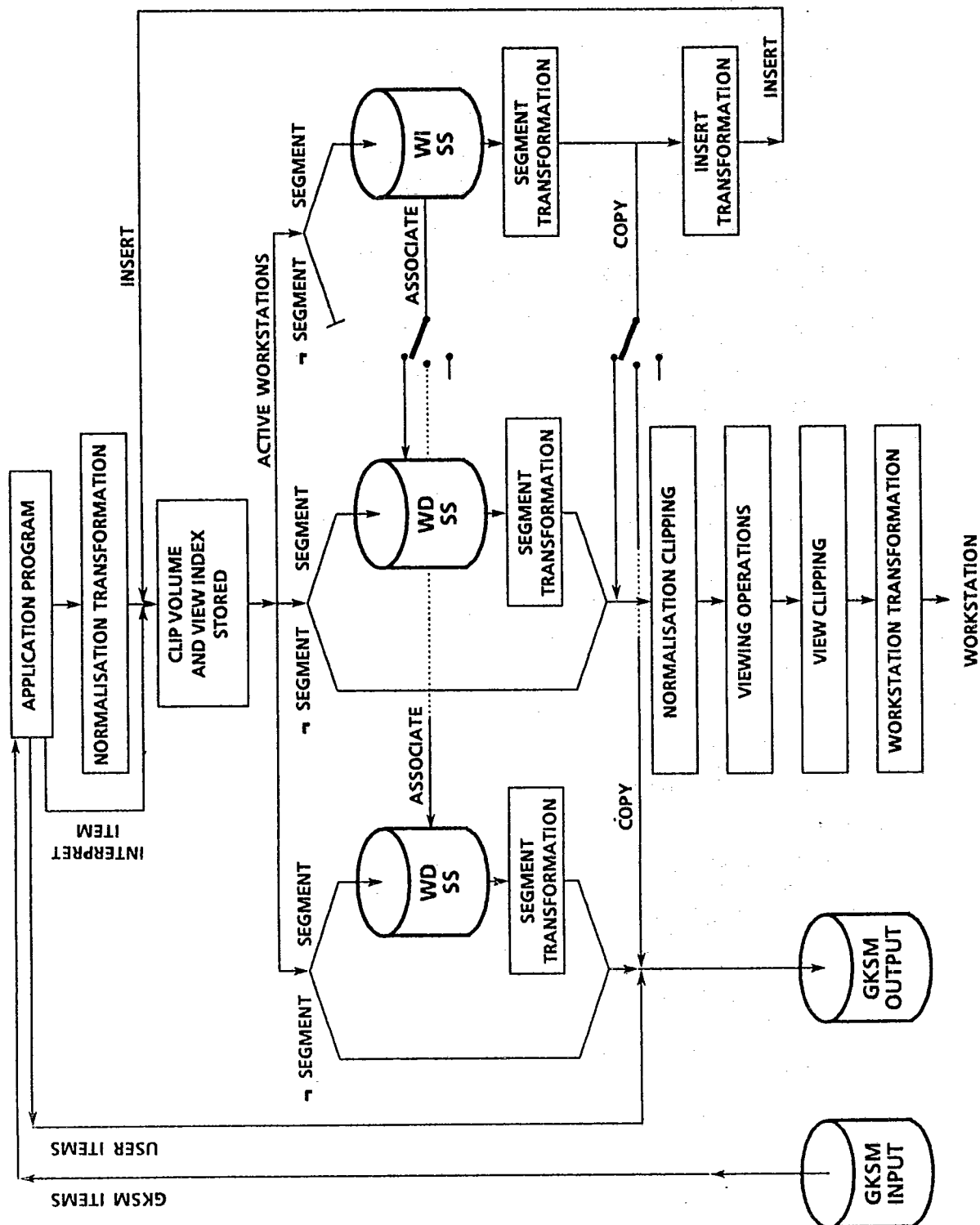


Figure 9R. Dataflow chart for graphical output in GKS-3D

4.6.4R 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. Conceptually, the input model is 3D, providing DC3 coordinates at the operator interface. However, two locator functions are provided, LOCATOR and LOCATOR 3, the former providing a shorthand form analogous to that defined for output primitives (see 4.8P).

To return to the application program a position in world coordinates, it is necessary to transform the input data from DC3 to WC3 by applying the inverse transformation pipeline. For the LOCATOR 3 function, the input position is returned to the application program as an (X,Y,Z) world coordinate point along with the number of the normalization and the viewing transformation used. For the LOCATOR function, the input position is returned only as an (X,Y) world coordinate point along with number of the normalization transformation used, the Z world coordinate value and viewing transformation are not passed to the application. There is only one pipeline for both the LOCATOR 3 and LOCATOR functions.

The normalization transformation is selected by comparing the position with the viewports of all normalization transformations in a sequence according to the setting of the viewport input priority. The same procedure is applied as in GKS except that a position may lie outside the unit cube of NDC3 in which case normalization transformation 0 is applied. The viewing transformation is selected similarly by comparing the position with the projection viewports of all viewing transformations on the workstation in a sequence according to the setting of the viewing transformation input priority. The projection viewport of viewing transformation 0 will always contain the position.

Setting the input priority of any view transformation to a priority lower than that of view transformation 0 effectively disables input from that view transformation. (This is because all valid locator values are within view transformation 0 which takes precedence).

In EVENT mode, the generation and later translation of locator input cannot be considered an atomic action. Therefore it is advisable for the application program not invoke any function that effects the inverse transformation while a locator EVENT is outstanding.

4.6.5R Transformation of stroke input

Similar considerations apply to the transformation of stroke input as apply to locator input, with the complication that more than one point is involved. As for locator input, two functions are provided, STROKE and STROKE 3, with the treatment of information passed back to the application program equivalent to that for LOCATOR and LOCATOR 3.

If SAMPLE mode stroke input is used, successive STROKE or STROKE 3 function calls may provide different world coordinate point sets caused by using different transformations in the procedure above.

DD 128:1986

In EVENT mode, stroke input has the same potential race condition as locator input, and the application program should be aware of the possible conflicts.

[ISO page 41]

This publication is a Draft for Development and is therefore not to be regarded as a British Standard. See page 5.

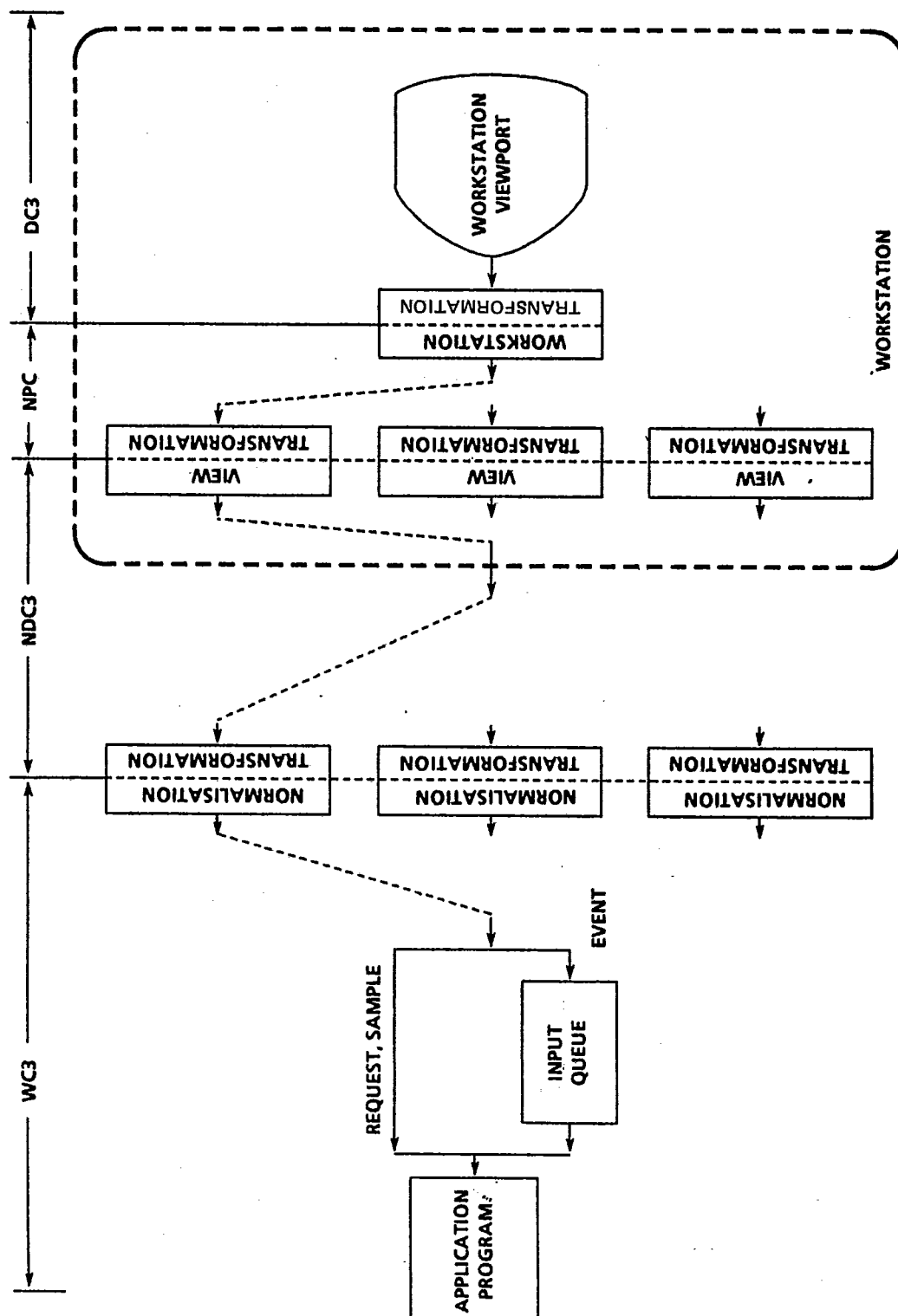


Figure 10R. Data flow chart for locator input

4.6.6 Viewing

Viewing is the mechanism for the realization of three dimensional graphical information on a workstation. Figure 17 shows the position of viewing in the graphical output pipeline. In GKS-3D, a view index is bound to each graphical output primitive on creation. This view index is used at each active workstation to access a view bundle from a view bundle table. The view bundle specifies a mapping from NDC3 to NPC. The purpose of this mapping is to do a projection for those workstations that require a projection for the display of three dimensional graphical information (e.g. raster display). A workstation may use all, part or none of the viewing mechanism dependent on its display requirements.

View bundle table entries are ordered in a list defined by the view transformation input priority. The view table entry with view index zero is assigned the highest view transformation input priority initially.

Several views may be defined on each workstation using SET VIEW REPRESENTATION. The default view (view index 0) is predefined and does a parallel projection mapping the NDC3 unit cube onto the NPC unit cube. The parameters for the default view cannot be altered. SET VIEW REPRESENTATION defines the view bundle parameters for a specific workstation and view index. Since the specification of a view on a workstation may be deferred, SET VIEW REPRESENTATION only sets the 'requested' view parameters in the view bundle table. The 'current' view parameters in the view bundle table continue to hold their previous set values. When the display is updated the current values are set to the requested values.

SET VIEW INDEX sets the 'current view index' in the GKS-3D state list and thus defines the value of view index which will be bound to succeeding graphical output primitives. Since the view representation for a particular view index is workstation dependent, the appearance of a particular graphical output primitive may differ between active workstations. Figure 18 shows the effect of a primitive with view index 2 bound to it passing through the viewing stage of the output pipeline with multiple active workstations.

The parameters to SET VIEW REPRESENTATION define two transformations and a view volume. The view volume is defined by front and back planes and a view window, each of which may be clipped to independently of the others. Figure 19 shows the three stages in the viewing mechanism defined by the parameters to SET VIEW REPRESENTATION.

The view transformation is specified by a 4×4 'view matrix' supplied by the application program. A utility function (EVALUATE VIEW MATRIX) has been provided to compute a 4×4 matrix from parameters for a model supported by GKS-3D.

The view volume is specified in the view reference coordinate (VRC) system produced by the view transformation. Figures 20 and 21 show the view volumes associated with parallel and perspective views respectively.

The front plane, back plane and view plane are planes parallel to the UV plane of the VRC system. They are specified as signed distances along the N axis by the parameters 'front plane distance', 'back plane distance' and 'view plane distance' respectively.

The view window is specified as U and V values on the view plane by the parameters (UMIN, UMAX, VMIN, VMAX). The projection reference point (PRP), in VRC space, orients the projectors defining the view volume. For a parallel projection the projectors are all parallel to the vector joining the PRP and the centre of the view window. The view volume is thus a parallelepiped. For a perspective projection the projectors all pass through the PRP. The view volume is thus a truncated frustrum (if the PRP is between the front and back planes the resultant view volume is a double pyramid). The PRP may not lie in the view plane.

The projection transformation is specified by a mapping from the view volume onto the projection viewport, such that the part of the front plane defined by the view volume coincides with the projection viewport ZMAX plane and the part of the back plane defined by the view volume coincides with the projection viewport ZMIN plane. The projection viewport is a rectangular parallelepiped in NPC space. The NPC system is a normalized intermediate coordinate system allowing hidden line/hidden surface removal algorithms to interface to the viewing system (see 4.5.7). Although NPC space conceptually extends beyond $[0,1] \times [0,1] \times [0,1]$, the part of NPC space in which the projection viewport needs to be located and which can be displayed at a workstation is restricted to this range. A separate NPC space exists for each active workstation and each is used conceptually to assemble the viewed graphical output primitives into a single displayable volume.

If invalid view parameters are provided to SET VIEW REPRESENTATION then the default parameters associated with view index 0 are used with the view index specified.

The default table entries for each view are:

a) View matrix:	1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1
b) Projection type:	PARALLEL
c) Projection reference point:	(0.5,0.5,1)
d) View window limits:	0,1,0,1
e) View plane distance:	0
f) Back plane distance:	0
g) Front plane distance:	1
h) Projection viewport limits:	0,1,0,1,0,1
i) Window clipping indicator:	CLIP
j) Back clipping indicator:	CLIP
k) Front clipping indicator:	CLIP

EVALUATE VIEW MATRIX is a utility function that allows the view matrix to be computed from a model supported by GKS-3D. The model allows an arbitrary positioning and orientation of the VRC system. The parameters to EVALUATE VIEW MATRIX specify a 'view reference point', a 'view plane normal' and a 'view up vector'. A 'coordinate switch' is provided to allow specification of these parameters in either WC3 or NDC3. The function returns a 'view matrix' and an 'error indicator'. If the input parameters are not specified correctly the appropriate error number is returned in the 'error indicator' and does not generate an error in the error file.