

INTERNATIONAL STANDARD

ISO/IEC
9636-3

First edition
1991-12-15

Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification —

Part 3: Output

*Technologies de l'information — Infographie — Interfaces pour
l'infographie — Spécifications fonctionnelles —*

Partie 3: Sortie



Reference number
ISO/IEC 9636-3:1991(E)

AVA Documents
Document Sharing Hub

Contents

	Page
Foreword	vii
Introduction	viii
1 Scope	1
2 Normative references	2
3 Concepts	3
3.1 Introduction	3
3.2 General output concepts	3
3.2.1 Types of graphic primitive functions	3
3.2.2 Attributes and controls	4
3.2.3 Output states	4
3.3 Individual and bundled attribute values	5
3.3.1 Introduction	5
3.3.2 Modes of attribute specification and selection	6
3.4 Colour	6
3.4.1 Direct and indexed modes	6
3.4.2 Background colour	8
3.5 Graphic objects	9
3.5.1 Compound objects	9
3.5.2 Global and local attributes	9
3.5.3 Detail of graphic object formation	9
3.6 Clipping associated with graphic objects	10
3.6.1 Rendering pipelines for clipping	11
3.7 Line primitives	13
3.7.1 Line functions	13
3.7.2 Line attributes	13
3.7.3 Line geometry	14
3.7.4 Line clipping	14
3.7.5 Allowed latitude	15
3.8 Marker primitive	15
3.8.1 Marker function	15
3.8.2 Marker attributes	15
3.8.3 Marker geometry	16
3.8.4 Marker clipping	16
3.9 Text primitives	17
3.9.1 Text functions	17
3.9.2 Usage of text functions (compound text)	17
3.9.3 Text attributes	17
3.9.4 Text geometry	19
3.9.5 Text clipping	27
3.9.6 Text fonts and character sets	28
3.9.7 Errors in TEXT OPEN state	28
3.9.8 Allowed latitude	28
3.10 Fill primitives	29
3.10.1 Fill functions	29
3.10.2 Fill attributes	29
3.10.3 Fill geometry	31
3.10.4 Fill clipping	32
3.10.5 Closed figures	33
3.10.6 Allowed latitude	37
3.11 Image primitive	37
3.11.1 Image function	38

© ISO/IEC 1991

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

	3.11.5	Allowed latitude	38
3.12		Generalized Drawing Primitives	39
	3.12.1	GDP function	39
3.13		Inquiry	39
	3.13.1	State lists and description tables	39
3.14		Retrieval	39
	3.14.1	Retrieval of text extent	39
4		Interactions with other parts of ISO/IEC 9636	41
	4.1	Interactions with all other parts of ISO/IEC 9636	41
	4.1.1	Character set and font selection	41
	4.2	Interactions with ISO/IEC 9636-2 (Control)	41
	4.2.1	Effect of INITIALIZE	41
	4.2.2	Effect of VDC Extent and VDC Type	41
	4.3	Interactions with ISO/IEC 9636-4 (Segments)	42
	4.3.1	CLIP RECTANGLE, CLIP INDICATOR, and COPY SEGMENT	42
	4.3.2	LINE WIDTH and EDGE WIDTH	42
	4.3.3	MARKER SIZE	42
	4.3.4	PICK IDENTIFIER	42
	4.3.5	Dynamic modification	42
	4.3.6	Segment open state	42
	4.4	Interactions with ISO/IEC 9636-5 (Input)	43
	4.5	Interactions with ISO/IEC 9636-6 (Raster)	43
	4.5.1	State related restrictions	43
	4.5.2	Interior style BITMAP	43
	4.5.3	Drawing modes	43
5		Abstract specification of functions	44
	5.1	Introduction	44
	5.1.1	Data types employed	44
	5.2	Graphic primitive functions	44
	5.2.1	POLYLINE	44
	5.2.2	DISJOINT POLYLINE	45
	5.2.3	CIRCULAR ARC 3 POINT	45
	5.2.4	CIRCULAR ARC CENTRE	45
	5.2.5	CIRCULAR ARC CENTRE REVERSED	46
	5.2.6	ELLIPTICAL ARC	47
	5.2.7	CONNECTING EDGE	47
	5.2.8	POLYMARKER	48
	5.2.9	TEXT	48
	5.2.10	RESTRICTED TEXT	49
	5.2.11	APPEND TEXT	50
	5.2.12	POLYGON	50
	5.2.13	POLYGON SET	51
	5.2.14	RECTANGLE	51
	5.2.15	CIRCLE	52
	5.2.16	CIRCULAR ARC 3 POINT CLOSE	52
	5.2.17	CIRCULAR ARC CENTRE CLOSE	53
	5.2.18	ELLIPSE	53
	5.2.19	ELLIPTICAL ARC CLOSE	54
	5.2.20	CELL ARRAY	54
	5.2.21	GENERALIZED DRAWING PRIMITIVE (GDP)	55
	5.3	Attribute functions	56
	5.3.1	LINE BUNDLE INDEX	56
	5.3.2	LINE TYPE	56
	5.3.3	LINE WIDTH	57
	5.3.4	LINE COLOUR	57
	5.3.5	LINE CLIPPING MODE	57
	5.3.6	MARKER BUNDLE INDEX	57
	5.3.7	MARKER TYPE	58
	5.3.8	MARKER SIZE	58
	5.3.9	MARKER COLOUR	58
	5.3.10	MARKER CLIPPING MODE	59

5.3.11	TEXT BUNDLE INDEX	59
5.3.12	TEXT FONT INDEX	59
5.3.13	TEXT PRECISION	59
5.3.14	CHARACTER EXPANSION FACTOR	60
5.3.15	CHARACTER SPACING	60
5.3.16	TEXT COLOUR	60
5.3.17	CHARACTER HEIGHT	60
5.3.18	CHARACTER ORIENTATION	61
5.3.19	TEXT PATH	61
5.3.20	TEXT ALIGNMENT	61
5.3.21	CHARACTER SET INDEX	62
5.3.22	ALTERNATE CHARACTER SET INDEX	62
5.3.23	CHARACTER CODING ANNOUNCER	62
5.3.24	FILL BUNDLE INDEX	63
5.3.25	INTERIOR STYLE	63
5.3.26	FILL COLOUR	63
5.3.27	HATCH INDEX	63
5.3.28	PATTERN INDEX	64
5.3.29	FILL REFERENCE POINT	64
5.3.30	PATTERN SIZE	64
5.3.31	EDGE BUNDLE INDEX	65
5.3.32	EDGE TYPE	65
5.3.33	EDGE WIDTH	65
5.3.34	EDGE COLOUR	66
5.3.35	EDGE CLIPPING MODE	66
5.3.36	EDGE VISIBILITY	66
5.4	General attribute and output control functions	66
5.4.1	CLIP INDICATOR	66
5.4.2	CLIP RECTANGLE	67
5.4.3	LINE WIDTH SPECIFICATION MODE	67
5.4.4	EDGE WIDTH SPECIFICATION MODE	67
5.4.5	MARKER SIZE SPECIFICATION MODE	68
5.4.6	COLOUR SELECTION MODE	68
5.4.7	COLOUR VALUE EXTENT	68
5.4.8	BACKGROUND COLOUR	69
5.4.9	AUXILIARY COLOUR	69
5.4.10	TRANSPARENCY	69
5.4.11	COLOUR TABLE	70
5.4.12	LINE REPRESENTATION	70
5.4.13	MARKER REPRESENTATION	70
5.4.14	TEXT REPRESENTATION	71
5.4.15	FILL REPRESENTATION	72
5.4.16	EDGE REPRESENTATION	72
5.4.17	DELETE BUNDLE REPRESENTATION	73
5.4.18	ASPECT SOURCE FLAGS	73
5.4.19	PATTERN TABLE	74
5.4.20	DELETE PATTERN	74
5.4.21	FONT LIST	75
5.4.22	CHARACTER SET LIST	75
5.4.23	SAVE PRIMITIVE ATTRIBUTES	75
5.4.24	RESTORE PRIMITIVE ATTRIBUTES	76
5.4.25	DELETE PRIMITIVE ATTRIBUTE SAVE SET	77
5.4.26	BEGIN FIGURE	77
5.4.27	END FIGURE	77
5.4.28	NEW REGION	78
5.5	Retrieval functions	78
5.5.1	GET TEXT EXTENT	78
6	Output inquiry functions	80
6.1	Introduction	80
6.1.1	Data types employed	80
6.1.2	Validity of returned information	80

6.2	Primitive support description table	80
6.2.1	INQUIRE PRIMITIVE SUPPORT LEVELS	80
6.2.2	LOOKUP GDP SUPPORT	81
6.2.3	INQUIRE GDP ATTRIBUTES	81
6.3	Line description table	81
6.3.1	INQUIRE LINE CAPABILITY	81
6.3.2	INQUIRE LIST OF AVAILABLE LINE TYPES	81
6.3.3	INQUIRE LIST OF AVAILABLE SCALED LINE WIDTHS	82
6.4	Marker description table	82
6.4.1	INQUIRE MARKER CAPABILITY	82
6.4.2	INQUIRE LIST OF AVAILABLE MARKER TYPES	82
6.4.3	INQUIRE LIST OF AVAILABLE SCALED MARKER SIZES	82
6.5	Text description table	83
6.5.1	INQUIRE TEXT CAPABILITY	83
6.5.2	INQUIRE LIST OF AVAILABLE CHARACTER SETS	83
6.5.3	INQUIRE LIST OF AVAILABLE TEXT FONTS	83
6.5.4	INQUIRE FONT CAPABILITIES	84
6.5.5	INQUIRE LIST OF AVAILABLE CHARACTER EXPANSION FACTORS	84
6.5.6	INQUIRE LIST OF AVAILABLE CHARACTER SPACINGS	84
6.5.7	INQUIRE LIST OF AVAILABLE CHARACTER HEIGHTS	85
6.5.8	INQUIRE LIST OF AVAILABLE CHARACTER ORIENTATIONS	85
6.6	Fill description table	85
6.6.1	INQUIRE FILL CAPABILITY	85
6.6.2	INQUIRE LIST OF AVAILABLE HATCH STYLES	86
6.7	Edge description table	86
6.7.1	INQUIRE EDGE CAPABILITY	86
6.7.2	INQUIRE LIST OF AVAILABLE EDGE TYPES	86
6.7.3	INQUIRE LIST OF AVAILABLE SCALED EDGE WIDTHS	87
6.8	Output control description table	87
6.8.1	INQUIRE COLOUR CAPABILITY	87
6.8.2	INQUIRE CIE CHARACTERISTICS	87
6.8.3	INQUIRE MAXIMUM NUMBER OF SIMULTANEOUSLY SAVED ATTRIBUTE SETS	87
6.8.4	INQUIRE ARRAY OF SUPPORTED CHARACTER CODING ANNOUNCERS	88
6.9	Line attribute state list	88
6.9.1	INQUIRE LINE ATTRIBUTES	88
6.9.2	INQUIRE LIST OF LINE BUNDLE INDICES	88
6.9.3	INQUIRE LINE REPRESENTATION	88
6.10	Marker attribute state list	89
6.10.1	INQUIRE MARKER ATTRIBUTES	89
6.10.2	INQUIRE LIST OF MARKER BUNDLE INDICES	89
6.10.3	INQUIRE MARKER REPRESENTATION	89
6.11	Text attribute state list	90
6.11.1	INQUIRE TEXT ATTRIBUTES	90
6.11.2	INQUIRE LIST OF TEXT BUNDLE INDICES	90
6.11.3	INQUIRE TEXT REPRESENTATION	90
6.12	Fill attribute state list	91
6.12.1	INQUIRE FILL ATTRIBUTES	91
6.12.2	INQUIRE PATTERN DIMENSIONS	91
6.12.3	INQUIRE PATTERN	91
6.12.4	INQUIRE LIST OF PATTERN INDICES	92
6.12.5	INQUIRE LIST OF FILL BUNDLE INDICES	92
6.12.6	INQUIRE FILL REPRESENTATION	92
6.13	Edge attribute state list	92
6.13.1	INQUIRE EDGE ATTRIBUTES	92
6.13.2	INQUIRE LIST OF EDGE BUNDLE INDICES	93
6.13.3	INQUIRE EDGE REPRESENTATION	93
6.14	General attributes and output control state list	93
6.14.1	INQUIRE OUTPUT STATE	93
6.14.2	INQUIRE OBJECT CLIPPING	94
6.14.3	INQUIRE LIST OF ATTRIBUTE SET NAMES IN USE	94
6.14.4	INQUIRE COLOUR STATE	94

	6.14.5	INQUIRE LIST OF COLOUR TABLE ENTRIES	94
	6.14.6	INQUIRE FONT LIST	95
	6.14.7	INQUIRE CHARACTER SET LIST	95
	6.14.8	LOOKUP ASPECT SOURCE FLAGS	95
7		CGI description tables and state lists	96
	7.1	Description tables	96
	7.1.1	Primitive support	96
	7.1.2	Attributes	97
	7.1.3	Font characteristics	100
	7.1.4	Output control	100
	7.2	State lists	101
	7.2.1	Attributes	101
	7.2.2	General attributes and output control	105
A		Formal grammar of the functional specification	106
B		Output and attribute errors	131
C		Guidelines for CGI implementors	133
D		Parameterization of circular and elliptical arcs	137
E		Use of POLYGON SET and closed figures	138
F		Character sets and coding	142
G		Colour value extent	146
H		Example use of character orientation	147

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 9636-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

ISO/IEC 9636 consists of the following parts, under the general title *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification*:

- *Part 1: Overview, profiles, and conformance*
- *Part 2: Control*
- *Part 3: Output*
- *Part 4: Segments*
- *Part 5: Input and echoing*
- *Part 6: Raster*

Annexes A and B form an integral part of this part of ISO/IEC 9636. Annexes C, D, E, F, G, and H are for information only.

Introduction

This part of ISO/IEC 9636 describes the functions needed for generating and displaying graphical output.

The functional capability incorporated in this part of ISO/IEC 9636 is concerned with graphic primitives and their associated attributes, control over the graphic object output pipeline, and the rendering of graphic objects. It should at least be read in conjunction with the overview in ISO/IEC 9636-1, and the general control functions detailed in ISO/IEC 9636-2.

The functional capability described in this part of ISO/IEC 9636 applies to CGI Virtual Devices of class OUTPUT and OUTIN.

Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices (CGI) – Functional specification –

Part 3: Output

1 Scope

This part of ISO/IEC 9636 establishes those functions of the Computer Graphics Interface concerned with output primitives and associated attributes and controls for creating graphical pictures.

This part of ISO/IEC 9636 is part 3 of ISO/IEC 9636, and should be read in conjunction with ISO/IEC 9636-1 and ISO/IEC 9636-2. The relationship of this part of ISO/IEC 9636 to the other parts of ISO/IEC 9636 is described in ISO/IEC 9636-1 and in clause 4.

The functionality described in this part of ISO/IEC 9636 pertains to OUTPUT and OUTIN classes of CGI Virtual Device.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9636. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9636 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

- ISO 646 : 1983 *Information processing – ISO 7-bit coded character set for information interchange.*
- ISO 2022 : 1986 *Information processing – ISO 7-bit and 8-bit coded character sets – Code extension techniques.*
- ISO 8632-1 : 1987 *Information processing systems – Computer graphics – Metafile for the storage and transfer of picture description information (CGM) – Part 1: Functional specification.*
- ISO/IEC 9541-1 : -¹⁾ *Information technology – Font information interchange – Part 1: Architecture.*
- ISO/IEC 9541-2 : -¹⁾ *Information technology – Font information interchange – Part 2: Interchange format.*
- ISO/IEC 9541-3 : -¹⁾ *Information technology – Font information interchange – Part 3: Glyph shape representation.*
- ISO/IEC 9592-1 : 1989 *Information processing systems – Computer graphics – Programmer's Hierarchical Interactive Graphics System (PHIGS) – Part 1: Functional description.*
- ISO/IEC 9636-1 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 1: Overview, profiles, and conformance.*
- ISO/IEC 9636-2 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 2: Control.*
- ISO/IEC 9636-4 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 4: Segments.*
- ISO/IEC 9636-5 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 5: Input and echoing.*
- ISO/IEC 9636-6 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 6: Raster.*
- ISO/IEC 9637-1 : -¹⁾ *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding — Part 1: Character encoding.*
- ISO/IEC 9637-2 : -¹⁾ *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding — Part 2: Binary encoding.*
- ISO/IEC TR 9973 : 1988 *Information processing — Procedures for registration of graphical items.*

¹⁾ To be published.

3 Concepts

3.1 Introduction

This part of ISO/IEC 9636 covers the device-independent graphic object output functionality of the CGI. It covers primitive functions, attributes, object formation and subsequent processing, and related control and inquiry functionality. This functionality is divided into the following areas:

- *Graphic primitive functions*, which describe the geometry of the components of a picture in the CGI.
- *Attribute functions*, which set modal values in state lists that are used to determine certain properties (including visual aspects) of these geometric picture components.
- *General attribute and output control functions*, which specify the modes of operation of certain other functions, control some aspects of the device's operation with respect to graphic objects and attribute functions, and which provide facilities for the construction of compound objects.
- *Retrieval function*, which returns information useful for the positioning of text objects.
- *Output inquiry functions*, which provide access to the description tables and state lists concerned with output and attributes.

3.2 General output concepts

3.2.1 Types of graphic primitive functions

The CGI defines graphic primitive functions for the specification of the geometry of the components of a picture.

These graphic primitive functions, grouped according to primitive type: line, marker, text, fill, and image, are as follows:

<i>Line functions:</i> POLYLINE DISJOINT POLYLINE CIRCULAR ARC 3 POINT CIRCULAR ARC CENTRE CIRCULAR ARC CENTRE REVERSED ELLIPTICAL ARC CONNECTING EDGE	<i>Fill functions:</i> POLYGON POLYGON SET RECTANGLE CIRCLE CIRCULAR ARC 3 POINT CLOSE CIRCULAR ARC CENTRE CLOSE ELLIPSE ELLIPTICAL ARC CLOSE
<i>Marker function:</i> POLYMARKER	<i>Text functions:</i> TEXT APPEND TEXT RESTRICTED TEXT
<i>Image function:</i> CELL ARRAY	<i>Generalized drawing primitive function:</i> GENERALIZED DRAWING PRIMITIVE (GDP)
NOTE – GDP does not have an explicit primitive type, but rather an instance of GDP may generate a primitive of one of the above primitive types depending on the specific GDP identifier.	

See ISO/IEC 9636-1, 5.2.1 for an overall description of graphic primitives in relationship to the CGI Graphic Object Pipeline.

3.2.2 Attributes and controls

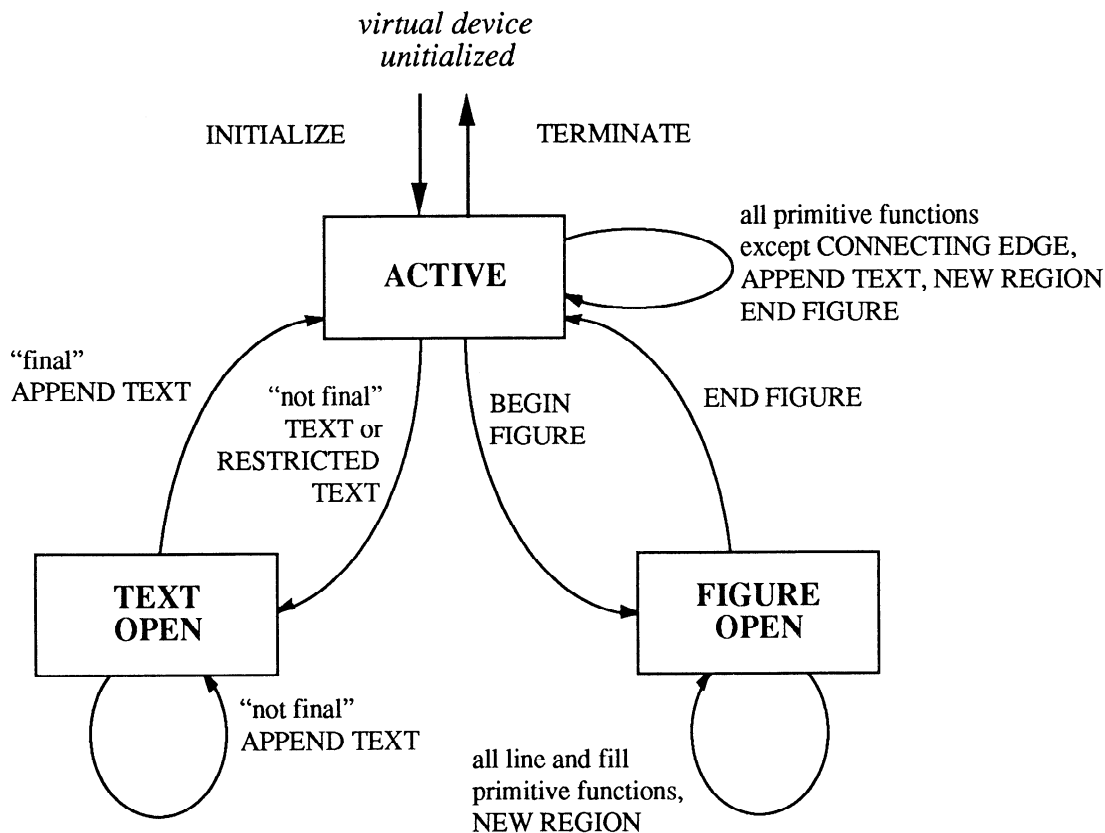
Primitives have attribute values associated with them as they pass through the Graphic Object Pipeline. Once a set of attribute values has been associated with a primitive the complete entity is referred to as a graphic object. Some of these attribute values determine the appearance of primitives when they are rendered as graphic objects. See ISO/IEC 9636-1, 5.2.1 for an overview of where attribute values are associated and used within the Graphic Object Pipeline.

Control functions are also defined in this part of ISO/IEC 9636 and provide control over other operations of the Graphic Object Pipeline, e.g. colour table. Some of these control functions also set corresponding entries in CGI state lists, but these control values are not associated with graphic primitives.

3.2.3 Output states

The Output State entry in the General Attributes and Output Control State List applies to the functions defined in this part of ISO/IEC 9636. The Output State may have only one of the values ACTIVE, TEXT OPEN, or FIGURE OPEN, the normal state being ACTIVE. It is set to state TEXT OPEN during the construction of a compound text object and to state FIGURE OPEN during the construction of a closed figure object. Figure 1 illustrates the output states of this part of ISO/IEC 9636 and indicates which functions result in a state transition.

State restrictions are defined in Output State TEXT OPEN for all primitive functions which do not contribute to compound text (see table 2). In particular, in Output State TEXT OPEN, all graphic primitive functions except APPEND TEXT are prohibited. Similarly, in Output State FIGURE OPEN, primitive functions which do not contribute to the construction of closed figures (i.e. all but line and fill type primitives) are prohibited (see table 3). In Output State ACTIVE, primitive functions which are relevant only in the state TEXT OPEN and FIGURE OPEN are prohibited (see table 1). There are no state restrictions applicable to attribute setting functions in any of these states.



NOTE – Whether or not a particular GDP is allowed in any of the output states is dependent on the specification of the GDP and its defined effect on the Output State entry in the General Attributes and Output Control State List.

Figure 1 – Output state diagram (for primitive functions)

Table 1 – Functions Not Allowed in Output State ACTIVE

APPEND TEXT	CONNECTING EDGE
END FIGURE	NEW REGION

Table 2 – Functions Not Allowed in Output State TEXT OPEN

POLYLINE	CIRCULAR ARC CENTRE
DISJOINT POLYLINE	CIRCULAR ARC CENTRE REVERSED
CIRCULAR ARC 3 POINT	ELLIPTICAL ARC
CONNECTING EDGE	
POLYGON	CIRCULAR ARC 3 POINT CLOSE
POLYGON SET	CIRCULAR ARC CENTRE CLOSE
RECTANGLE	ELLIPSE
CIRCLE	ELLIPTICAL ARC CLOSE
BEGIN FIGURE	END FIGURE
NEW REGION	
TEXT	RESTRICTED TEXT
POLYMARKER	CELL ARRAY

Table 3 – Functions Not Allowed in Output State FIGURE OPEN

TEXT	POLYMARKER
RESTRICTED TEXT	CELL ARRAY
APPEND TEXT	BEGIN FIGURE

3.3 Individual and bundled attribute values

3.3.1 Introduction

The appearance of a graphic object is determined by a number of attribute values associated with the object. These attribute values control both the geometric and non-geometric aspects of the object. In addition, there is an attribute value which is used to identify an object or group of objects in a segment (see ISO/IEC 9636-4 for the definition of pick identifier).

These attribute values are set modally and have entries in primitive type specific state lists. Individual specification of these attributes is provided by separately defined CGI functions. A subsequent change of the state list entry for an attribute will not affect the value associated with any existing graphic objects. Attribute values associated from such state list entries are referred to as individual attribute values.

Some attribute values may, in addition to this modal specification, be grouped into bundle entries in various bundle tables. Bundle tables are defined for line, marker, text, fill, and edge. A particular bundle entry can be selected for use with an object via a bundle index attribute. There is a bundle index attribute value for each primitive type (with the exception of the image type).

Whether the value for a particular attribute is a directly associated modal value or whether it is indirectly referenced by means of a bundle index attribute value and subsequently obtained from the referenced bundle is determined by a further set of attributes, called aspect source flags (ASFs). Each ASF may take either the value INDIVIDUAL or BUNDLED and allows the distinction to be made for each of the attribute values which may be bundled.

A number of bundles are predefined when the CGI Virtual Device is initialized. Each such predefined bundle shall be visually distinguishable from that of the other predefined bundles for a given primitive type when all aspect source flags are BUNDLED. These and other bundles may subsequently be redefined by the client. Bundles may be deleted by the client. However, the predefined bundle with index 1 cannot be deleted.

If the values of a bundle are changed and that bundle is already in use, a modification of the rendered picture may result. Whether such modification of the rendered picture occurs immediately on execution of the changing function or at some later time is determined by the Implicit Segment Regeneration Mode (see ISO/IEC 9636-4) and is dependent on whether the Virtual Device has segment capabilities. The Dynamic Modification Accepted For entries in the output description tables of an implementation indicate which changes may be performed immediately or may lead to subsequent regeneration.

3.3.2 Modes of attribute specification and selection

In CGI state lists, the values for attributes controlled by specification or selection modes are stored in the mode in which they were last specified (or the mode corresponding to the default value) along with the value of the corresponding mode. The corresponding attributes in the bundle tables are also stored in the mode in which they were specified, along with the specification mode. When the attribute value is associated with a primitive, during object formation, the mode in which it was set is also associated. The stored mode acts as a data type identification, and is used for that purpose in the inquiry functions based on the state list. These stored values are thereafter independent of the current specification or selection mode on the General Attributes and Output Control State List.

3.4 Colour

3.4.1 Direct and indexed modes

The CGI provides two mechanisms for colour selection:

- Direct, in which the colour is defined by providing values for the normalized weights of the RGB components, and
- Indexed, in which the colour is defined by an index into a table of colour values.

Selection of one of these mechanisms may be achieved using the function COLOUR SELECTION MODE. It is possible to have both direct and indexed colour in use simultaneously in the same picture on devices which implement direct colour selection for object colour attributes.

3.4.1.1 Direct colour and COLOUR VALUE EXTENT

For direct colour specification, the CGI uses the RGB additive colour model. Each colour specifier is a three-tuple of values providing the normalized weight of the red, green, and blue components of the desired colour.

The integer components of a client specified CD value are normalized to a three-tuple of abstract RGB colour values, each of which is a real number in the range [0.0, 1.0]. The normalization also has the property that any 3-tuple with 3 identical components, (x,x,x), represents equal weights of the red, green, and blue components. For any given component, one end of the range (0.0) indicates that none of that component is included, and the other end (1.0) indicates that the maximum intensity of that component is included in the colour, with an infinite number of component values in between. (0,0,0) thus represents black, (1,1,1) represents white, and (x,x,x) with x between 0 and 1 represents a grey.

The abstract minimum colour value of (0,0,0) is represented by (min_red, min_green, min_blue) and the abstract maximum colour value of (1,1,1) is represented by (max_red, max_green, max_blue). These minimum and maximum values are specified with the COLOUR VALUE EXTENT function which defines a linear mapping of these values onto the device's abstract colour range.

Although CGI supports only the RGB colour model, it provides entries in the Output Control Description Table for constants which are necessary for conversion of colour values between the CIE colour model and RGB colour model. A client may inquire these values and use them in performing the colour model conversion. (See ISO/IEC 9592-1, annex I.)

3.4.1.2 Indexed colour and COLOUR TABLE

The colour table used with indexed colour selection is a contiguous set of colour values indexed from the range [0, colour table size - 1]. It is initialized in the following manner: index 0 contains an implementation-dependent background colour; index 1 and greater contain implementation-dependent foreground colours. The COLOUR TABLE function is provided for changing the contents of the colour table by specifying the colour by RGB values as described above. The COLOUR TABLE function may be used at various times throughout the specification of a picture. However, whether changes in the colour table

affect the appearance of any rendered graphic objects that use the affected indices is indicated by the Dynamic Modification Accepted For Colour Table entry in the Output Control Description Table.

3.4.1.3 Gamma correction

There is no gamma correction nor specification of the source colours implied for the tri-stimulus values for direct colour specification. This degree of control over colour may be available on an implementation-dependent basis through the use of ESCAPE functions.

3.4.1.4 Colour and monochromatic devices

All devices capable of displaying output are considered to have at least two colours – the background and the foreground. The drawing medium will always count as one colour for the Number of Simultaneously Available Direct Colours, Number of Simultaneously Available Indexed Colours, and Number of Available Colours entries in the Output Description Table. Therefore, none of the colour counts may be less than two.

Interpretation of the Number of Simultaneously Available Direct Colours or the Number of Simultaneously Available Indexed Colours, when either is greater than two, depends on whether the device is monochromatic. If the Monochromatic Device entry in the Output Control Description Table is NO, then the device is capable of rendering at least two different foreground colours (varying in either hue or saturation) but not differing exclusively in intensity (such as light green, dark green). If the Monochromatic Device entry is YES, then the interpretation of the Number of Simultaneously Available Direct Colours when colour selection mode is DIRECT and the Number of Simultaneously Available Indexed Colours when colour selection mode is INDEXED is taken to mean the number of steps in a single colour intensity scale whose low value is closest to the default background colour (or drawing medium for those devices without background colour capability) and whose high value has maximum contrast relative to the default background colour.

3.4.1.5 Transparency and auxiliary colour

Various graphic primitives have “holes” which may or may not be rendered. Examples of such “holes” include gaps in dashed lines or edges, the background of a character cell, and the spaces between lines in a hatched fill primitive.

Two of the general attributes which are associated with graphic primitives as they enter the graphic object pipeline are TRANSPARENCY and AUXILIARY COLOUR. The associated TRANSPARENCY attribute value determines whether the drawing of the “holes” occurs. If this attribute value is TRANSPARENT, those portions of the object which are considered to be “holes” will not be drawn; hence, potentially affected portions of the graphic picture remain unchanged. If this attribute value is OPAQUE, those portions of the object which are considered to be “holes” will be drawn; hence, affected portions of the graphic picture will be modified. The associated AUXILIARY COLOUR attribute value is used in determining the colour of the “holes”. The DRAWING MODE attribute associated with the object is not applied (see 3.4.1.6) until after the application of TRANSPARENCY and AUXILIARY COLOUR.

When TRANSPARENCY is OPAQUE, non-solid (e.g. dot-dash) lines or edges, text, and hatched interiors for fill objects are rendered as two separate point sets in abstract DC space: one with the normal associated colour attribute values and the other with the associated auxiliary colour attribute value.

3.4.1.6 Overwrite capability, colour realization, and DRAWING MODE

Due to inherent technology differences, the physical devices used as the display for CGI Virtual Devices differ in their behaviour when something is rendered in the same physical location on the drawing surface as previously rendered output or displayed on the display surface at the same location as previously displayed output.

Raster technology devices can overwrite the contents of the drawing surface, so that each object rendered “in front of” previously rendered ones (see the lower right-hand picture shown in figure 2). Vector devices cannot usually do this: the Colour Overwrite entry in the Output Control Description Table indicates whether the Virtual Device has this capability.

Output rendered on the drawing surface will ultimately be displayed on the display surface; it may be mixed additively or subtractively on the display surface with which previously displayed output (see the lower left-hand picture shown in figure 2). The Colour Realization entry in the Output Control Description Table indicates which colour mixing is used.

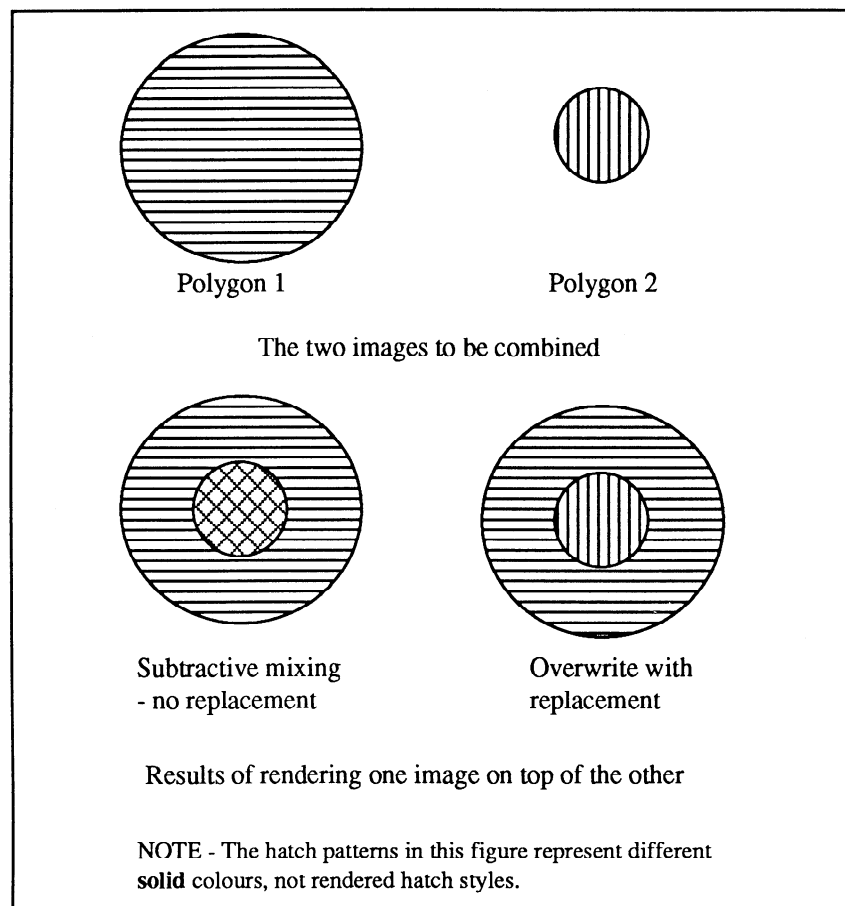


Figure 2 – Overwrite capability

Device offering functionality defined in ISO/IEC 9636-6 provide an additional means of controlling colour mixing on the drawing surface – the drawing mode. For these devices, the Colour Overwrite Capability is typically YES, and hence the DRAWING MODE function fully controls the colour mixing on the drawing surface.

3.4.2 Background colour

The drawing surface is set to the background colour by PREPARE DRAWING SURFACE or by a regeneration prior to any rendering. The CGI provides several ways of specifying the background colour. The implemented ways are indicated by the Background Colour Capability entry in the Output Control Description Table.

Four levels of background colour capability are defined, as follows:

- NONE:** The device is not capable of changing the background colour, e.g. this is a property of the media on which the picture is drawn. (In this case, the BACKGROUND COLOUR function need not be provided.)
- INDEX 0:** The device has a background colour which is always associated with colour index 0. Changing the setting of colour table entry 0 will perform the desired change. (In this case, the BACKGROUND COLOUR function need not be provided.)
- INDEXED:** The device has a background colour which is always associated with a colour index, which can be changed by the client. In this case, the BACKGROUND COLOUR function which accepts only CI values shall be provided. The default colour index for background colour is 0.
- FULL:** The device has the capability of using either an indexed colour for background colour or a direct colour which is not associated with any colour table entry. In this case, the BACKGROUND COLOUR function

with full support of the CO data type shall be provided. The default is specified in INDEXED mode with value 0.

3.5 Graphic objects

This part of ISO/IEC 9636 defines the functions which create graphic primitives in the Graphic Object Pipeline. A graphic primitive becomes a graphic object when the appropriate attribute values are associated with the primitive. Graphic objects are passed down the Graphic Object Pipeline for further refinement, storage, and rendering.

An outline of this process is given in ISO/IEC 9636-1, 5.2.1. The following sub-clauses provide further details of this process.

3.5.1 Compound objects

In addition to the graphic primitive functions listed in 3.2.1, this part of ISO/IEC 9636 defines functions that provide a client of the CGI with means to create compound objects from several of the other graphic primitives. The following classes of compound object are defined: compound text and closed figures. The functions that may be used to specify compound objects are listed in table 4.

Table 4 – Contributing Primitives to Compound Objects

Object Class	Initiating Function	Functions allowed in open state	Other Functions	Closing Function
Compound Text	TEXT (Note 1) RESTRICTED TEXT (Note 1)	APPEND TEXT		APPEND TEXT (Note 2)
Closed Figure	GDP (Note 4) BEGIN FIGURE	GDP (Note 4) Line Functions Fill Functions (Note 3) GDP (Note 4)	NEW REGION	GDP (Note 4) END FIGURE
NOTES 1) Final/not-final flag is NOT FINAL; the primitive defines the reference point of the entire compound text object; the text primitive is entered in the text buffer. 2) Final/not-final flag is FINAL; the text primitive is entered in the text buffer before the compound text object is closed. 3) All primitives of the identified primitive types may be included. 4) Whether a GDP may contribute to compound text or closed figures, and how it interacts with the Output State is specified with the definition of the GDP in the International Register of Graphical Items or in the implementation documentation.				

3.5.2 Global and local attributes

For the purposes of compound object formation, a further classification of attributes into either global or local attributes is introduced.

Global attributes are those whose values are associated with a completed compound object (e.g. fill colour, drawing mode). Global attributes apply to a compound object as a whole.

Local attributes are those whose values are separately associated with the graphic primitives that make up a compound object (e.g. text colour, edge type, drawing mode). Local attributes apply separately to the component graphic primitives of a compound object.

3.5.3 Detail of graphic object formation

In ISO/IEC 9636-1, 5.2.1 the formation of graphic objects is described as occurring in a single step. This subclause further defines the details of this formation to include the formation of compound graphic objects (see figure 3).

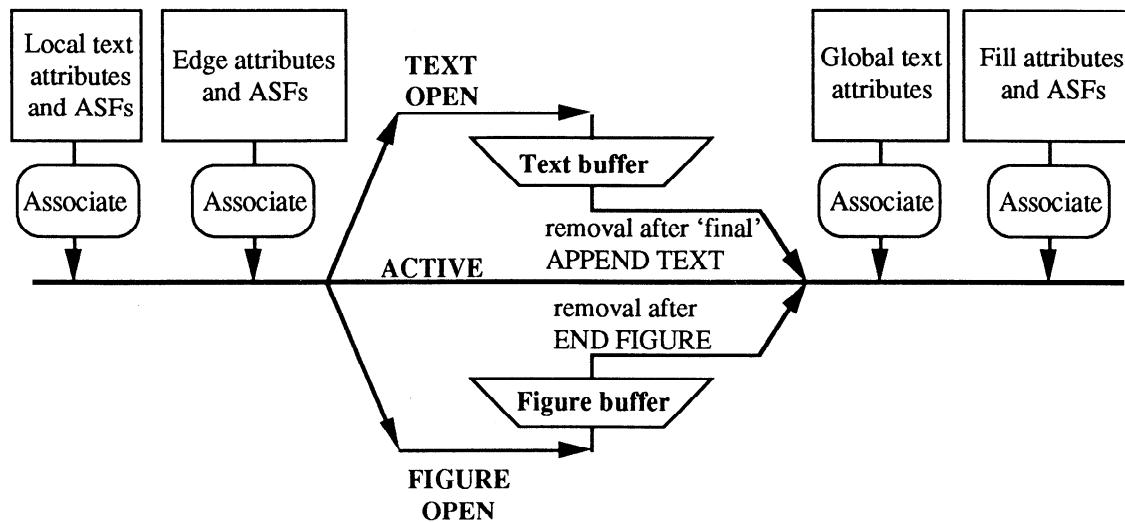


Figure 3 – Detail of the graphic object pipeline for compound objects

While a compound object is being formed, graphic primitives have their local attribute values associated and are then conceptually stored in an intermediate buffer. When the function that closes the compound object is executed, the contents of the intermediate buffer are combined and the graphic object is completed by association of the relevant global attribute values.

The process of compound object formation is not affected by, and does not itself affect, any process further down the pipeline defined in ISO/IEC 9636-1, 5.2.1.

3.6 Clipping associated with graphic objects

The general attribute values, CLIP RECTANGLE and CLIP INDICATOR, along with any applicable object clipping mode attribute values, are associated with a primitive as it enters the Graphic Object Pipeline. These attribute values control the subsequent form of clipping applied to the graphic object. The CLIP RECTANGLE is associated as the clip region.

If the CLIP INDICATOR is OFF, the effective clip region is the VDC range. If the CLIP INDICATOR is ON, the effective clip region is the intersection of the clip region associated with the graphic object and the VDC extent.

There are three different object clipping modes which specifically apply to lines, markers, and edges. These are maintained in CGI state lists and are manipulated by means of the functions LINE CLIPPING MODE, EDGE CLIPPING MODE, and MARKER CLIPPING MODE. When the CLIP INDICATOR associated with a graphic object is ON, only those parts of a graphic object that are considered inside the effective clip region are rendered. The object clipping modes allow the client to be more precise in how clipping is applied to graphic objects.

An object clipping mode may be either LOCUS, SHAPE, or LOCUS THEN SHAPE. Conceptually, a locus is a mathematical object like a point or line segment, while a shape is an area in two-dimensional space. Loci are zero-dimensional or one-dimensional subsets of real-valued two-dimensional space. For markers and text, they are points. For lines, they are the individual line segments or portions of arcs. Shapes reflect the realization of geometric attributes such as LINE TYPE and LINE WIDTH and are generally two-dimensional subsets of real-valued space.

When a width or size specification mode is SCALED, the rendering of shape proceeds in DC space after application of the VDC-to-Device Mapping. When a width or size specification mode is VDC, the abstract rendering of shape proceeds in VDC space before application of segment and copy transformations and the VDC-to-Device Mapping. This abstract shape is then transformed point by point and mapped to DC space for shape clipping.

The effect of each object clipping mode is as follows:

Clipping associated with graphic objects

Concepts

- LOCUS clipping is applied for each portion of a graphic object based on its mathematical location and is independent of the area it will occupy after rendering. For example, no portion of a line segment is rendered if the entire ideal mathematical line segment lies outside the effective clip region (even if its line width would carry some portion of the rendering of it into the effective clip region). No portion of a marker is rendered if its location point lies outside the effective clip region.

If LOCUS clipping is used, the rendering is applied to the clipped locus of the graphic object. The resulting rendered shape areas may therefore extend outside the effective clip region.

- SHAPE clipping is applied after the abstract rendering of shape in real DC space. The two-dimensional point set associated with the graphic object is intersected with the effective clip region, which has been transformed to abstract DC space.
- LOCUS THEN SHAPE clipping allows the client to specify that both LOCUS and SHAPE clipping be applied to graphic objects as described above. In this case, the rendered shape will not extend outside the effective clip region.

Fill, text, and image objects do not have associated object clipping mode values (although the edge of a fill object does). Clipping for fill and image objects is always consistent with SHAPE clipping.

For text objects, the type of clipping is determined by the associated text precision value:

- For STRING precision text, clipping may proceed, on a per string basis, in a manner consistent with LOCUS clipping.
- For CHARACTER precision text, clipping may proceed, on a per character basis, in a manner consistent with LOCUS clipping.
- For STROKE precision text, the clipping always proceeds in a manner consistent with SHAPE clipping.

Note that a Virtual Device is always allowed by ISO/IEC 9636 to use SHAPE clipping for any text precision.

Drawing Surface Clipping (see ISO/IEC 9636-2, 3.3.7) applies to all types of graphic objects. The drawing surface clipping always corresponds to SHAPE clipping, as described above.

3.6.1 Rendering pipelines for clipping

Figure 4 illustrates the effect that object clipping modes and specification modes have on the Graphic Object Pipeline presented in ISO/IEC 9636-1. Any conceptual changes to the pipeline apply on a per object basis; that is, each object passing down the pipeline may have different clipping modes and specification modes and therefore apparently traverse a different variation of the pipeline.

Conceptually, the object clipping modes determine the presence or absence within the pipeline of the *Apply Locus Clipping* and *Apply Shape Clipping* operations. For LOCUS THEN SHAPE clipping, both operations are present. For LOCUS clipping, only the *Apply Locus Clipping* operation is present. For SHAPE clipping, only the *Apply Shape Clipping* operation is present.

The effect of the width and size specification modes for lines, markers, and edges on the pipeline is dependent also on the corresponding clipping modes.

If the specification mode is VDC, the operation *Render Shape* takes place prior to the application of the associated transformation and the VDC-to-Device Mapping so that the VDC specification of the shape is properly transformed and mapped to device coordinates. In addition, if the object clipping mode is one of LOCUS or LOCUS THEN SHAPE, the transformed and clipped locus of a graphic object is transformed by the inverse associated transformation in order that the *Render Shape* operation is applied correctly. The associated transformation is then applied to the rendered shape of the object.

The combination VDC/SHAPE also applies to fill, text, and, image objects.

If the specification mode is SCALED, the operation *Render Shape* takes place following the application of the VDC-to-Device Mapping. In this case, when the object clipping mode includes LOCUS, the associated transformation is only applied once.

VDC sizes will change proportionally with the size of the device viewport (assuming that the VDC extent remains unchanged). VDC sizes are also affected by the copy and segment transformations. For SCALED sizes, the size remains fixed with respect to the drawing surface.

Concepts

Clipping associated with graphic objects

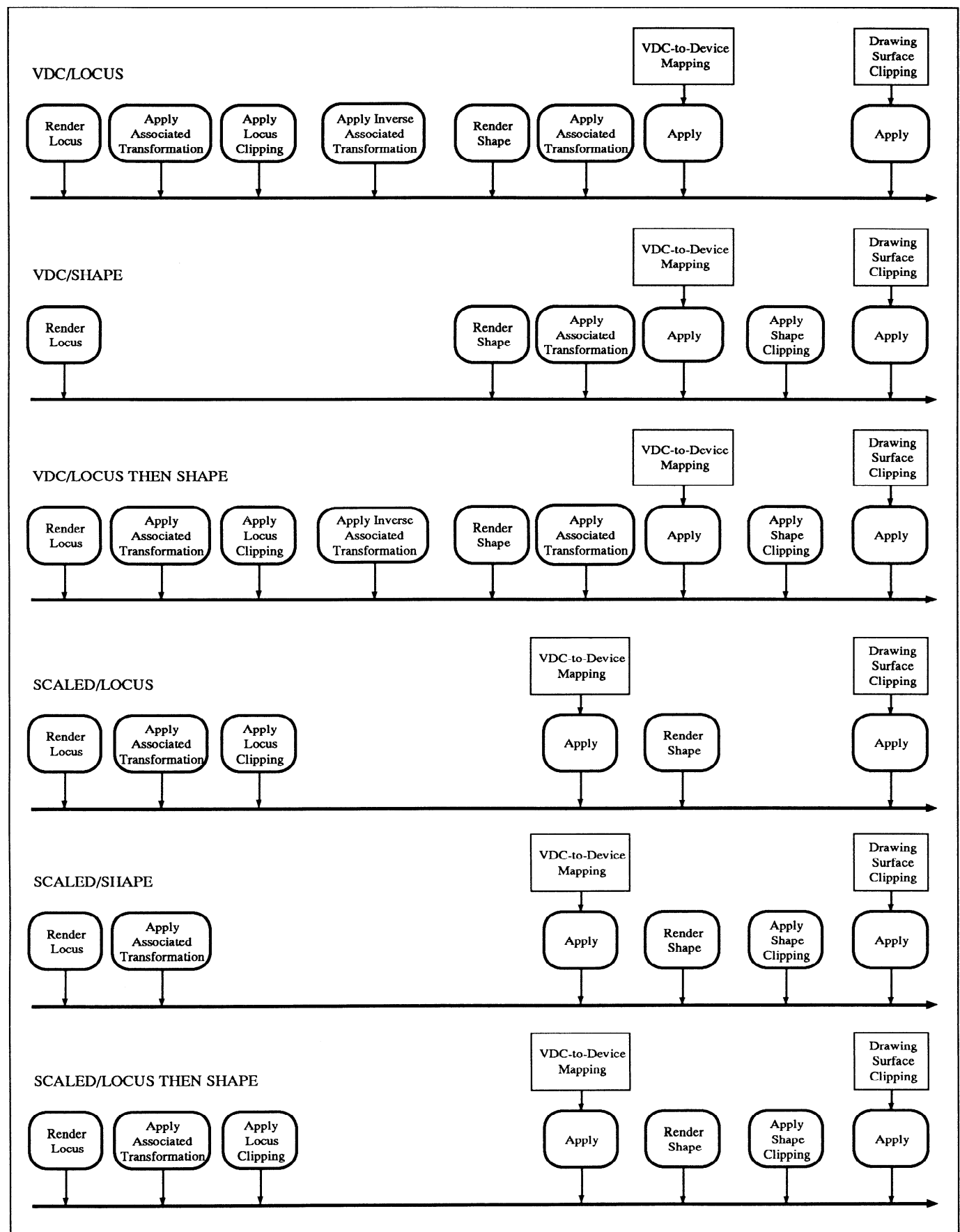


Figure 4 – Rendering pipelines for size specification mode/clipping mode combinations

3.7 Line primitives

3.7.1 Line functions

The CGI defines two general line functions, several line functions relating to circles and ellipses which generate line primitives, and a special line function which generates a line primitive that is only used within the FIGURE OPEN state.

POLYLINE	generates a set of connected straight line segments defined by a list of points, starting with the first point, drawing a line segment to each successive point, ending at the last point;
DISJOINT POLYLINE	generates a set of unconnected straight line segment defined by a list of point pairs, drawing from the first to the second, the third to the fourth, etc.;
CIRCULAR ARC CENTRE	these functions each generate a single circular arc; but provide alternative parameterizations of the arc; they are described in 5.2 (and in annex D);
CIRCULAR ARC CENTRE REVERSED	
CIRCULAR ARC 3 POINT	
ELLIPTICAL ARC	generates a single elliptical arc; the parameterization of the arc is described in 5.2 (and in annex D);
CONNECTING EDGE	used only in closed figure construction, this special line function adds an edge portion and an explicit boundary portion to the closed figure – it effectively converts what would have been the next implicit boundary portion to an explicit boundary portion. See 3.10.5.

3.7.2 Line attributes

The appearance of line primitives is described by specific line attributes and other general attributes.

LINE TYPE	determines the type of line (e.g. solid, dotted, or dashed), with which the shape of a line object is rendered;
SPECIFICATION MODE OF LINE WIDTH	determines whether the line width attribute value is interpreted as a DC scaling factor or a VDC width during rendering; see 3.7.3;
LINE WIDTH	determines the width of line with which the shape of a line object is rendered;
LINE COLOUR	determines the colour with which a line object is rendered. The colour selection mode in which the line colour was specified determines whether the LINE COLOUR attribute value is interpreted as a direct or indexed colour value during rendering. See 3.4 for a description of how colours are defined in the CGI;
LINE CLIPPING MODE	determines the type of object clipping to be performed during rendering; see 3.7.4;
LINE BUNDLE INDEX	determines the line bundle to be used during rendering when the corresponding line ASF attribute values are BUNDLED. The following line attribute values may be bundled: LINE TYPE, LINE WIDTH, and LINE COLOUR;
Line ASFs	determine which attribute values of the line bundle referenced by the LINE BUNDLE INDEX attribute are to be used during rendering when the corresponding ASF attribute value is BUNDLED. The following line ASFs are defined: LINE TYPE ASF, LINE WIDTH ASF, and LINE COLOUR ASF.

3.7.2.1 General attributes

CLIP INDICATOR	determines whether object clipping is to be applied to the line object;
CLIP RECTANGLE	used in determining the effective clip region to be applied to a line object if the associated CLIP INDICATOR is ON;

AUXILIARY COLOUR	determines the colour to render the gaps in a line for line types other than solid, if the TRANSPARENCY attribute value is OPAQUE. The colour selection mode in which the auxiliary colour was specified determines whether the AUXILIARY COLOUR attribute value is interpreted as a direct or indexed colour value during rendering;
TRANSPARENCY	determines whether gaps in the shape of a broken line type are rendered using the AUXILIARY COLOUR attribute value. If the TRANSPARENCY attribute value is TRANSPARENT, any previously drawn parts of the picture are unaffected in line gaps;
DRAWING MODE	determines how the rendered (and rastered) object is mixed with what already exists on the drawing surface (defined in ISO/IEC 9636-6);
PICK IDENTIFIER	(defined in ISO/IEC 9636-4).

3.7.3 Line geometry

The width of a line is determined by the LINE WIDTH attribute value. The LINE WIDTH attribute value is interpreted using the line width specification mode, which can take one of two values, SCALED or VDC. If the specification mode is SCALED, the width is interpreted as the nominal device-dependent line width multiplied by the given LINE WIDTH attribute value; thus, the line width is not affected by any transformations. If the specification mode is VDC, the width is interpreted in VDC units and is subject to segment and copy transformations and the VDC-to-Device Mapping; these transformations and the VDC-to-Device Mapping are applied after the abstract rendering of VDC line width and line type in VDC space.

3.7.3.1 Degeneracy of line objects

Degeneracy of line objects may arise either due to the client's specifications or unintentionally due to round-off error in transforming coordinate data.

If the Virtual Device cannot render a line of the exact specified width, the closest implemented width will be used. When line width is specified in VDC, then if, after its transformation to device coordinates, the resulting width is less than the Minimum Scaled Line Width, the Minimum Scaled Line Width is used; if the resulting width is greater than the Maximum Scaled Line Width, the Maximum Scaled Line Width is used. In general, if the abstract rendering of the locus of a line primitive degenerates to a zero-dimensional point set, at least a dot shall be rendered.

There are several ways in which an arc may degenerate. A circular arc 3 point or elliptical arc is non-degenerate if and only if the three specifying points are non-collinear. If the three points are all coincident, a dot is rendered. If the three points are collinear, the circular arc has zero curvature and a line is rendered from the starting point through the intermediate point to the ending point. If the points specifying an elliptical arc are collinear, an implementation-dependent straight line is rendered.

A circular arc with a radius of zero is rendered as a dot.

3.7.4 Line clipping

Line object clipping is controlled by the LINE CLIPPING MODE attribute, which can have one of three possible values: LOCUS, SHAPE, or LOCUS THEN SHAPE. However, object clipping only applies if the associated CLIP INDICATOR attribute is ON.

For LOCUS clipping, the mathematical locus of the line is clipped at its intersection with the effective clip region associated with the graphic object before shape rendering is applied. Hence, part of the shape of a line may appear outside the effective clip region.

For SHAPE clipping, the shape of the rendered line is clipped at its intersection with the effective clip region (i.e. nothing is drawn outside the effective clip region).

For LOCUS THEN SHAPE clipping, the mathematical locus of the line is clipped as with locus clipping, and then subsequently the rendered shape of the clipped locus is again clipped. Note that since the mathematical locus of the line may have changed as a result of locus clipping, subsequent shape rendering and clipping may produce a different appearance in a line from either of the other two clipping modes.

3.7.5 Allowed latitude

The following levels of Line Type Continuity are permitted for a CGI Virtual Device: RESTART, CONTINUOUS, and OTHER. The implemented level of support is indicated by the Line/Edge Type Continuity Capability entry in the Primitive Support Description Table.

- A level of RESTART indicates that the line type pattern is restarted at each vertex of a line object except for the last vertex;
- A level of CONTINUOUS indicates that the Virtual Device maintains the line type continuously across all vertices of a single line object except for the first and last vertices;
- A level of OTHER indicates that another method has been used (for example, guaranteeing some continuity and that end points of each line segment of line object are always drawn).

The preferred behaviour is CONTINUOUS.

3.8 Marker primitive

3.8.1 Marker function

The CGI provides a single marker function which generates a marker primitive:

POLYMARKER generates a symbol of a specific type at each point specified in a list of points.

3.8.2 Marker attributes

The appearance of marker primitives is described by specific marker attributes and other general attributes.

MARKER TYPE determines the type of marker (e.g. dot, plus, or asterisk), with which a marker object is rendered;

SPECIFICATION MODE OF MARKER SIZE determines whether the marker size attribute value is interpreted as a scaling factor or a VDC size during rendering. See 3.8.3;

MARKER SIZE determines the size of markers with which a marker object is rendered;

MARKER COLOUR determines the colour with which a marker object is rendered. The colour selection mode in which the marker colour was specified determines whether the MARKER COLOUR attribute value is interpreted as a direct or indexed colour value during rendering. See 3.4 for a description of how colours are defined in the CGI;

MARKER CLIPPING MODE determines the type of object clipping to be performed during rendering; see 3.8.4;

MARKER BUNDLE INDEX determines the marker bundle to be used during rendering when the corresponding marker ASF values are BUNDLED. The following marker attribute values may be bundled: MARKER TYPE, MARKER SIZE, and MARKER COLOUR;

Marker ASFs determine which attribute values of the marker bundle referenced by the MARKER BUNDLE INDEX attribute are to be used during rendering when the corresponding ASF attribute value is BUNDLED. The following marker ASFs are defined: MARKER TYPE ASF, MARKER SIZE ASF, and MARKER COLOUR ASF.

3.8.2.1 General attributes

CLIP INDICATOR determines whether object clipping is to be applied to the marker object;

CLIP RECTANGLE used in determining the effective clip region to be applied to a marker object if the associated CLIP INDICATOR is ON;

AUXILIARY COLOUR determines the colour to render the portions of a marker object, other than the foreground of a marker glyph, if the TRANSPARENCY attribute value is OPAQUE. The colour selection

	mode in which the auxiliary colour was specified determines whether the AUXILIARY COLOUR attribute value is interpreted as a direct or indexed colour value during rendering;
TRANSPARENCY	determines whether the portions of a marker object, other than the foreground of a marker glyph, are rendered using the AUXILIARY COLOUR attribute. If the TRANSPARENCY attribute value is TRANSPARENT, any previously drawn parts of the picture are unaffected by these portions of a marker object;
DRAWING MODE	determines how the rendered (and rastered) object is mixed with what already exists on the drawing surface (defined in ISO/IEC 9636-6);
PICK IDENTIFIER	(defined in ISO/IEC 9636-4).

3.8.3 Marker geometry

The locus of a marker is defined to be the single point which positions it. The rendering of the marker relative to its locus, which extends in two-dimensional space, is its shape.

The size of a marker is determined by the MARKER SIZE attribute value. The MARKER SIZE attribute value is interpreted using one of two specification modes, SCALED or VDC. If the specification mode is SCALED, the size is interpreted as the nominal device-dependent marker size multiplied by the given marker size attribute value; hence, the marker size is not affected by any transformations. If the specification mode is VDC, the size is interpreted in VDC units and is subject to segment and copy transformations and the VDC-to-Device Mapping.

The shape of markers is never affected by transformations (e.g. a circle used as a marker type shall always appear as a circle when rendered). Only the marker size (except in the case of marker type 1 (dot)) may be transformed. The transformation of marker size may be described as follows: consider the ratio of the length of a vector after transformation to its length before; choose a vector which maximizes this ratio. The resulting ratio is the Euclidean norm of the (2 x 2) transformation matrix. This ratio is used to scale the marker.

3.8.3.1 Degeneracy of marker objects

Degeneracy of marker objects may arise either due to the client's specifications or unintentionally due to round-off error in transforming coordinate data.

If the Virtual Device cannot render a marker of the exact specified size, the closest available size shall be used.

3.8.4 Marker clipping

Marker object clipping is controlled by the MARKER CLIPPING MODE attribute, which can have one of three possible values: LOCUS, SHAPE, or LOCUS THEN SHAPE. However, object clipping only applies if the associated CLIP INDICATOR attribute value is ON.

For LOCUS clipping, the mathematical loci of the markers (i.e. the defining points) are clipped at the intersection with the effective clip region associated with the graphic object before shape rendering is applied. Hence, part of the shape of a marker may appear outside the effective clip region. No portion of a marker symbol is rendered if its defining point lies outside of the effective clip region.

For SHAPE clipping, the shape of the rendered marker symbols is clipped at the intersection with the effective clip region (i.e. nothing is drawn outside the effective clip region). Portions of a marker symbol may appear inside the effective clip region even though that symbol's locating point is outside.

For LOCUS THEN SHAPE clipping, the mathematical loci of the markers are clipped as with locus clipping, and then subsequently the rendered shapes of the markers are again clipped as in SHAPE clipping to assure no portion of a marker shape is rendered outside the effective clip region.

3.9 Text primitives

3.9.1 Text functions

Text display in the CGI is produced by either a “simple” or “compound” text object. A compound text object is one that requires several text primitives to fully specify and is aligned and displayed as a single unit. The CGI provides the following text functions:

TEXT	generates a text primitive (which may begin a compound text object) aligned to a particular point;
RESTRICTED TEXT	generates a text primitive (which may begin a compound text object) aligned to a particular point and constrained within a given area specified as a parallelogram;
APPEND TEXT	generates a text primitive which appends the specified character string to the compound text object under construction.

3.9.2 Usage of text functions (compound text)

Each text function has a “not final/final” flag that indicates whether the generated text primitive contributes to the construction (i.e. as the start, continuation, or completion) of a compound text object.

If the Output State in the General Attributes and Output Control State List is ACTIVE and a TEXT or RESTRICTED TEXT function with the “not final/final” flag set to FINAL is invoked, a text primitive is generated which produces a simple text object, the appearance of which is controlled by the current text attributes.

If the Output State is ACTIVE and a TEXT or RESTRICTED TEXT function with the “not final/final” flag set to NOT FINAL is invoked, the Output State is set to TEXT OPEN and the construction of a compound text object is begun with the created text primitive. Subsequent APPEND TEXT functions contribute text primitives to the construction of the compound text object. A final APPEND TEXT function, that is, one with the “not final/final” flag set to FINAL, completes the compound text construction. The Output State is then set to state ACTIVE.

During the Output State TEXT OPEN, “local” text attributes from the Text Attributes State List and General Attributes and Output Control State List are associated with the individual text primitives generated by the various invocations of TEXT, RESTRICTED TEXT, and APPEND TEXT. The “global” text attributes from the Text Attributes State List are associated with the text object on completion of the compound text.

In general, a compound text object can only be rendered *after* its completion because of text alignment and the way in which attribute changes affect the definition of the text extent parallelogram. Note that the text extent parallelogram specified in the RESTRICTED TEXT function is not affected by any attribute changes.

3.9.2.1 Usage of GDP

A GDP which is defined as a text primitive may, in accordance with the GDP definition, cause transitions of the Output State to or from TEXT OPEN or contribute to compound text objects. Any variation or special handling applicable to a GDP in state TEXT OPEN shall be documented explicitly in the GDP description.

3.9.3 Text attributes

The appearance of all text primitives is described by specific text attributes and other general attributes.

3.9.3.1 Local text attributes

The following attributes are “local” for text primitives, that is, they are associated with text primitives which produce simple text objects and with individual text primitives which contribute to the construction of compound text objects.

CHARACTER HEIGHT	determines the height of the character body measured along the character up vector for the text object (see 3.9.4);
CHARACTER EXPANSION FACTOR	determines the width/height ratio of the characters of a text object (see 3.9.4);

Concepts

Text primitives

CHARACTER SPACING	determines the length of space between characters of a text object when rendered (see 3.9.4);
TEXT FONT INDEX	determines the font to use in rendering the text string (see 3.9.6);
ALTERNATE CHARACTER SET INDEX	determines an alternate character set which may be used within the text string (see 3.9.6);
CHARACTER SET INDEX	determines the normal character set used for characters in the text string (see 3.9.6);
TEXT COLOUR	determines the colour with which the text object is rendered. The colour selection mode in which the text colour was specified determines whether the TEXT COLOUR attribute value is interpreted as a direct or indexed colour value during rendering. See 3.4 for a description of how colours are defined in the CGI;
TEXT BUNDLE INDEX	determines the text bundle to be used during rendering if any of the text ASF attribute values are BUNDLED. The following text attribute values may be bundled: TEXT FONT INDEX, TEXT PRECISION, CHARACTER SPACING, CHARACTER EXPANSION FACTOR, and TEXT COLOUR;
Text ASFs	determine which attribute values of the text bundle referenced by the TEXT BUNDLE INDEX attribute are to be used during rendering when the corresponding ASF attribute value is BUNDLED. The following text ASFs are defined: TEXT FONT INDEX ASF, TEXT PRECISION ASF, CHARACTER SPACING ASF, CHARACTER EXPANSION FACTOR ASF, and TEXT COLOUR ASF;

3.9.3.2 Global text attributes

The following attributes are global for text primitives, that is, they are associated with text primitives which produce simple text objects and with complete compound text objects as a whole.

CHARACTER ORIENTATION	determines the character up vector and character base vector for the text object when rendered (see 3.9.4);
TEXT PRECISION	determines the requested rendering quality of the text object;
TEXT PATH	determines the “writing” direction of the characters of the text string with respect to character orientation (see 3.9.4);
TEXT ALIGNMENT	determines the position of the text extent parallelogram in relation to the text position (see 3.9.4);
CHARACTER CODING ANNOUNCER	determines the interpretation of the string parameter and techniques for switching character sets according to ISO 2022 (see 3.9.6).

3.9.3.3 General attributes

The following general attributes are associated with text primitives – AUXILIARY COLOUR and TRANSPARENCY as local attributes; CLIP INDICATOR, CLIP RECTANGLE, PICK IDENTIFIER, and DRAWING MODE as global attributes.

CLIP INDICATOR	determines whether object clipping is to be applied to the text object;
CLIP RECTANGLE	used in determining the effective clip region to be applied to a text object if the associated CLIP INDICATOR is ON;
AUXILIARY COLOUR	determines the colour to render the portions of the body of a character, other than the foreground of a character glyph, if the TRANSPARENCY attribute value is OPAQUE. The colour selection mode in which the auxiliary colour was specified determines whether the AUXILIARY COLOUR attribute value is interpreted as a direct or indexed colour value during rendering;
TRANSPARENCY	determines whether the portions of the body of a character, other than the foreground of a character glyph are rendered using the AUXILIARY COLOUR attribute value. If the TRANSPARENCY attribute value is TRANSPARENT, any previously drawn parts of the picture are unaffected by these portions of characters;

Text primitives**Concepts**

DRAWING MODE	determines how the rendered (and rastered) object is mixed with what already exists on the drawing surface (defined in ISO/IEC 9636-6);
PICK IDENTIFIER	(defined in ISO/IEC 9636-4).

3.9.4 Text geometry

The character body reference lines are illustrated in figure 5. The character body normally encloses all of the drawn parts (kerning excepted) of all characters in the font (that is, no descender extends lower than the bottom reference line, and no accent mark or oversized symbol extends higher than the top reference line). The left and right reference lines of the character body may be defined on a per character basis to accommodate variable widths and proportional spacing. The body exceeds the actual character symbol width and height as necessary to provide adequate white space between characters both vertically and horizontally.

NOTE – This insures that text is readable and adequately separated when adjacent character bodies are flush (i.e. when CHARACTER SPACING is 0) and multiline text can be created using continuous values of TEXT ALIGNMENT without causing vertical overlaps in a one-way output environment.

In some cases, a font may have been designed so that some elements (i.e. “bowls” and “arms”) of the font may extend beyond the character body horizontally; however, the RESTRICTED TEXT function still requires that all visible portions of the character be confined to the text extent parallelogram.

3.9.4.1 Character sizing and orientation

The CHARACTER HEIGHT attribute specifies the VDC distance between the capline and baseline of the font (see figures 5 and 6) measured along the character up vector. Note that if the character up vector is not perpendicular to the character base vector, the distance from the baseline to the capline measured perpendicular to the baseline will be less than the CHARACTER HEIGHT attribute. The CHARACTER HEIGHT attribute also affects the character width in such a way that the aspect ratio of the character body is not changed.

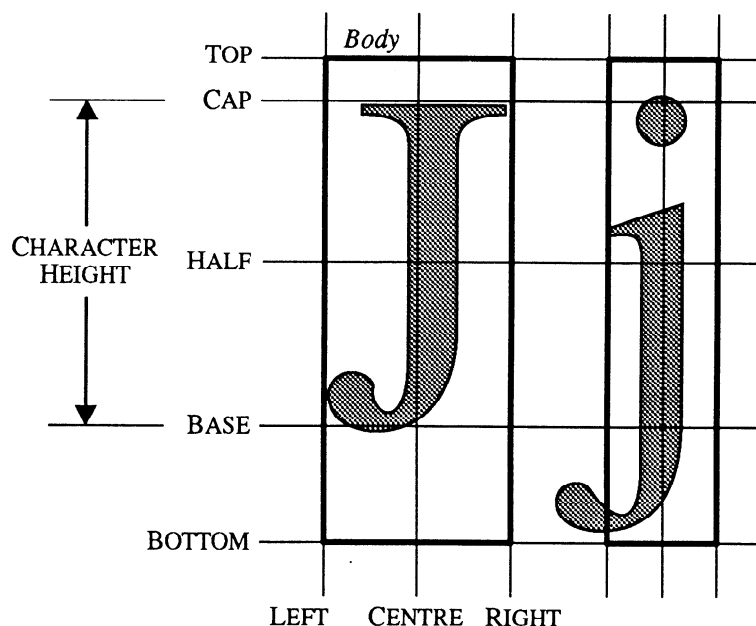


Figure 5 – Character body reference lines

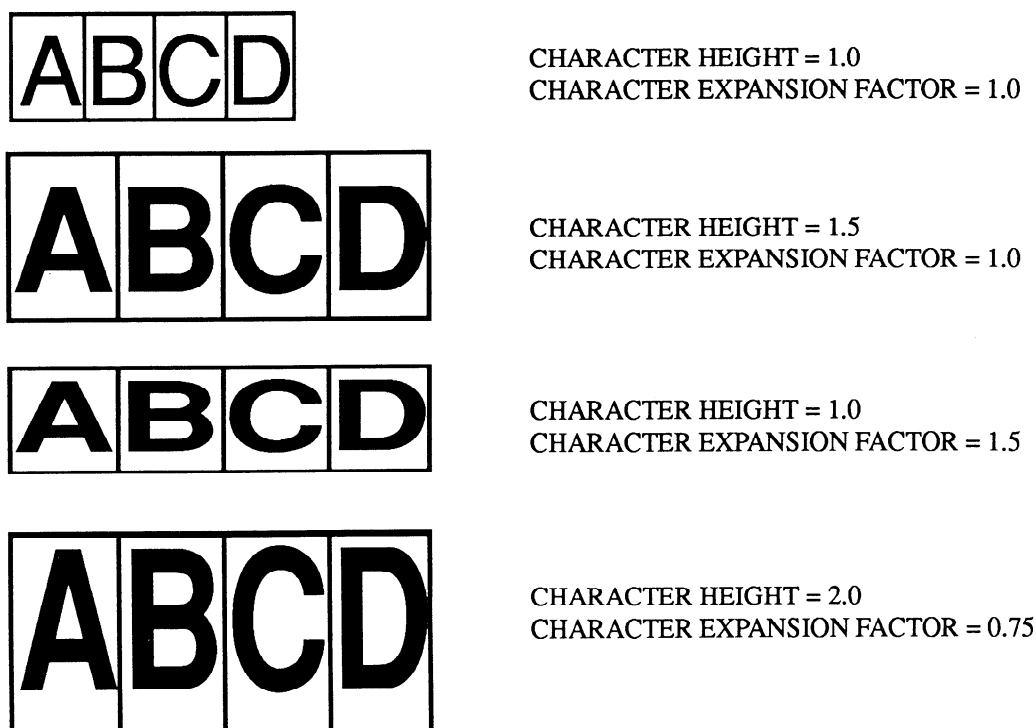


Figure 6 – CHARACTER HEIGHT and CHARACTER EXPANSION FACTOR

The CHARACTER EXPANSION FACTOR attribute specifies the deviation of the width to height ratio of the characters from the ratio indicated by the font designer (see figure 6).

The CHARACTER SPACING attribute specifies how much additional space is to be inserted between two adjacent character bodies (see figure 7). If the value of CHARACTER SPACING is zero, the character bodies are arranged flush with one another along the TEXT PATH with only the inter-character spacing designated by the font designer. If the value of CHARACTER SPACING is positive, additional space is inserted between character bodies. If the value of CHARACTER SPACING is negative, adjacent character bodies overlap although the character symbols themselves might not. Character spacing is specified as a fraction of the CHARACTER HEIGHT.

The CHARACTER ORIENTATION attribute specifies the character up vector and base vector, which determine the orientation and skew of the characters, and also determine the sense of RIGHT, LEFT, UP, and DOWN for TEXT PATH and TEXT ALIGNMENT (see figures 8 and 9).

The Virtual Device uses the ratio of the length of the transformed base vector to the length of the transformed up vector to scale the CHARACTER SPACING for text paths RIGHT and LEFT, and the CHARACTER EXPANSION FACTOR in all cases, before these are used to display the text.

3.9.4.2 Text path

TEXT PATH has the possible values of RIGHT, LEFT, UP, and DOWN. It specifies the writing direction of the text string (see figures 7 and 15).

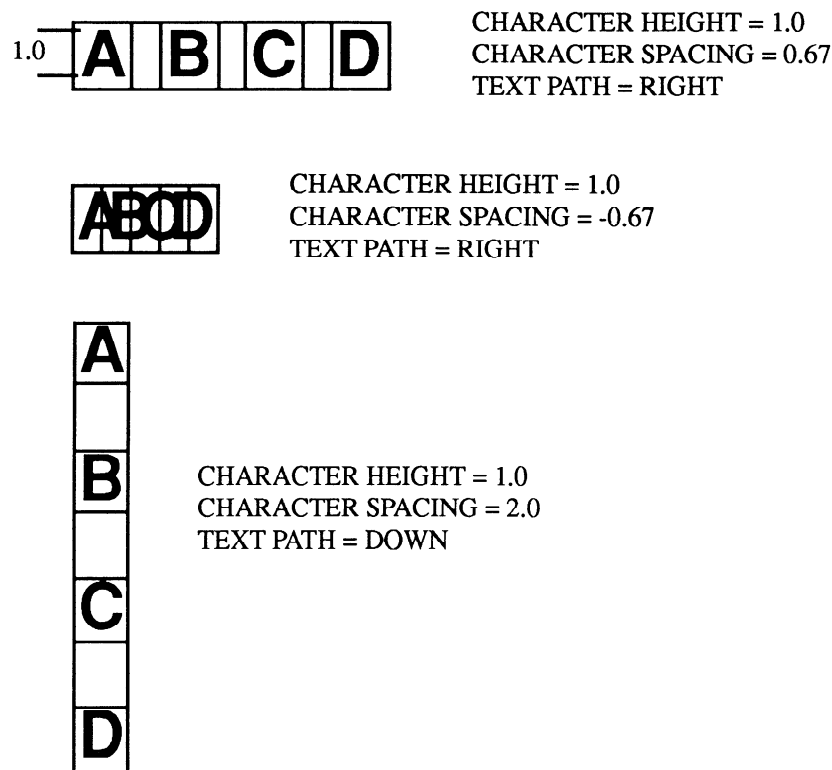


Figure 7 – CHARACTER SPACING

RIGHT means in the direction of the character base vector.

LEFT means 180° from the character base vector.

UP means in the direction of the character up vector.

DOWN means 180° from the character up vector.

For the UP and DOWN text path directions, the characters are arranged so that the centres of the character bodies are on a straight line in the direction of the character up vector. For the LEFT and RIGHT text path directions, the characters are arranged so that the baselines of the characters are on a straight line parallel to the direction of the character base vector.

NOTE – because of the effect on TEXT PATH, it is recommended that skewed character orientation vectors *not* be used to simulate slanted or italicized fonts when TEXT PATH is UP or DOWN.

These composition rules also apply when characters of different heights, expansion factors, or fonts are intermixed in a compound text object by means of attribute changes between a TEXT function and subsequent APPEND TEXT functions.

3.9.4.3 Text alignment

When aligning a text object, alignment of text is with respect to a text extent parallelogram relative to CHARACTER ORIENTATION, which is derived by joining the character bodies of the characters in the string according to the current status of the attributes and the composition rules described. The text extent parallelogram is the smallest parallelogram with sides parallel to the character orientation vectors and containing all of the character bodies in the text object.

The TEXT ALIGNMENT attribute controls the positioning of the text extent parallelogram in relation to the text position (see figures 10, 11, 12, and 13).

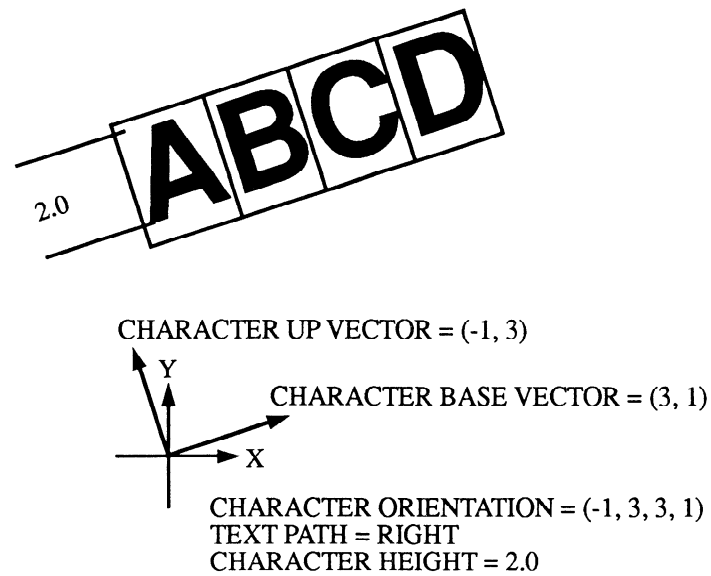


Figure 8 – CHARACTER ORIENTATION

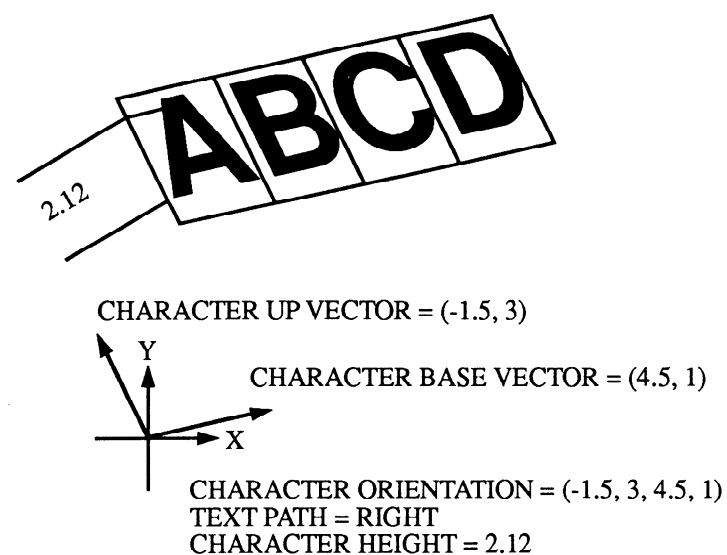


Figure 9 – CHARACTER HEIGHT and CHARACTER ORIENTATION with skewed orientation vectors

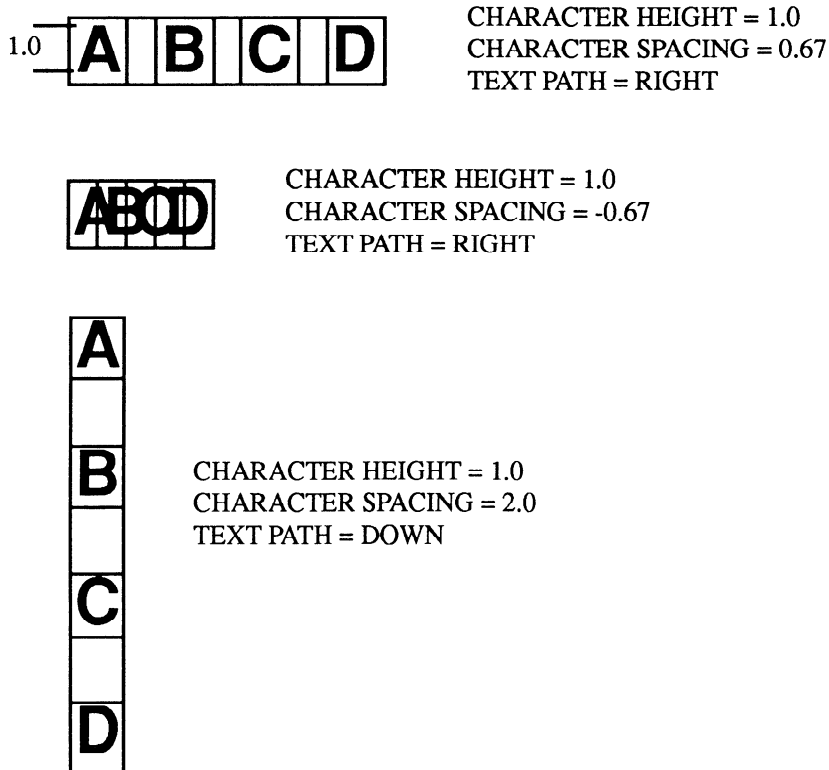


Figure 7 – CHARACTER SPACING

RIGHT means in the direction of the character base vector.

LEFT means 180° from the character base vector.

UP means in the direction of the character up vector.

DOWN means 180° from the character up vector.

For the UP and DOWN text path directions, the characters are arranged so that the centres of the character bodies are on a straight line in the direction of the character up vector. For the LEFT and RIGHT text path directions, the characters are arranged so that the baselines of the characters are on a straight line parallel to the direction of the character base vector.

NOTE – because of the effect on TEXT PATH, it is recommended that skewed character orientation vectors *not* be used to simulate slanted or italicized fonts when TEXT PATH is UP or DOWN.

These composition rules also apply when characters of different heights, expansion factors, or fonts are intermixed in a compound text object by means of attribute changes between a TEXT function and subsequent APPEND TEXT functions.

3.9.4.3 Text alignment

When aligning a text object, alignment of text is with respect to a text extent parallelogram relative to CHARACTER ORIENTATION, which is derived by joining the character bodies of the characters in the string according to the current status of the attributes and the composition rules described. The text extent parallelogram is the smallest parallelogram with sides parallel to the character orientation vectors and containing all of the character bodies in the text object.

The TEXT ALIGNMENT attribute controls the positioning of the text extent parallelogram in relation to the text position (see figures 10, 11, 12, and 13).

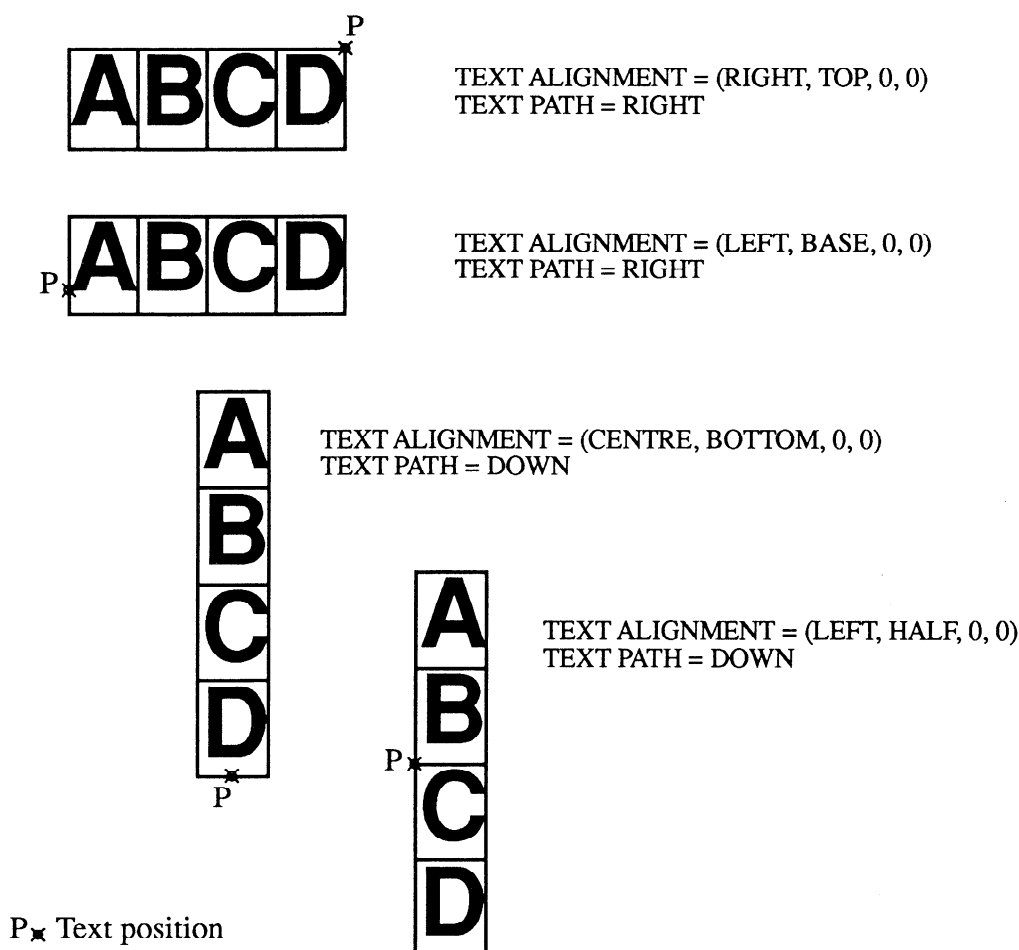


Figure 10 – Discrete text alignment

The horizontal alignment type of TEXT ALIGNMENT has five possible values: LEFT, CENTRE, RIGHT, NORMAL HORIZONTAL, and CONTINUOUS HORIZONTAL. If the horizontal alignment type is LEFT, the left side of the text extent parallelogram passes through the text position. Similarly, if the value is RIGHT, the right side of the text extent parallelogram passes through the text position. If the horizontal alignment type is CENTRE, the text position lies on a line parallel to and mid-way between the left and right sides of the text extent parallelogram. In this case, if TEXT PATH equals UP or DOWN, the straight line passing through the centrelines of the characters also passes through the text position.

If the horizontal alignment type is CONTINUOUS HORIZONTAL, the text position lies on a line parallel to the ends of the text extent parallelogram whose relative position with respect to the left and right ends of the text extent parallelogram is specified by the continuous horizontal alignment parameter. A value of 0.0 corresponds to the left end and a value of 1.0 corresponds to the right end. Negative values and values in excess of 1.0 are allowed. (See figure 11.)

The vertical alignment type of TEXT ALIGNMENT has seven possible values: TOP, CAP, HALF, BASE, BOTTOM, NORMAL VERTICAL, and CONTINUOUS VERTICAL. A vertical alignment type of TOP, CAP, HALF, BASE, or BOTTOM causes the text to be placed such that the corresponding reference line of one of the character bodies in the text string passes through the text position. In cases where this may be different for different characters, the ambiguity is resolved in accordance with tables 5 and 6.

If the vertical alignment type is CONTINUOUS VERTICAL, the text position lies on a line parallel to the top of the text extent parallelogram whose relative position with respect to the bottom and top of the text extent parallelogram is specified by the continuous vertical alignment parameter. A value of 0.0 corresponds to the bottom side and a value of 1.0 corresponds to the top side. Negative values and values in excess of 1.0 are allowed. (See figure 11.)

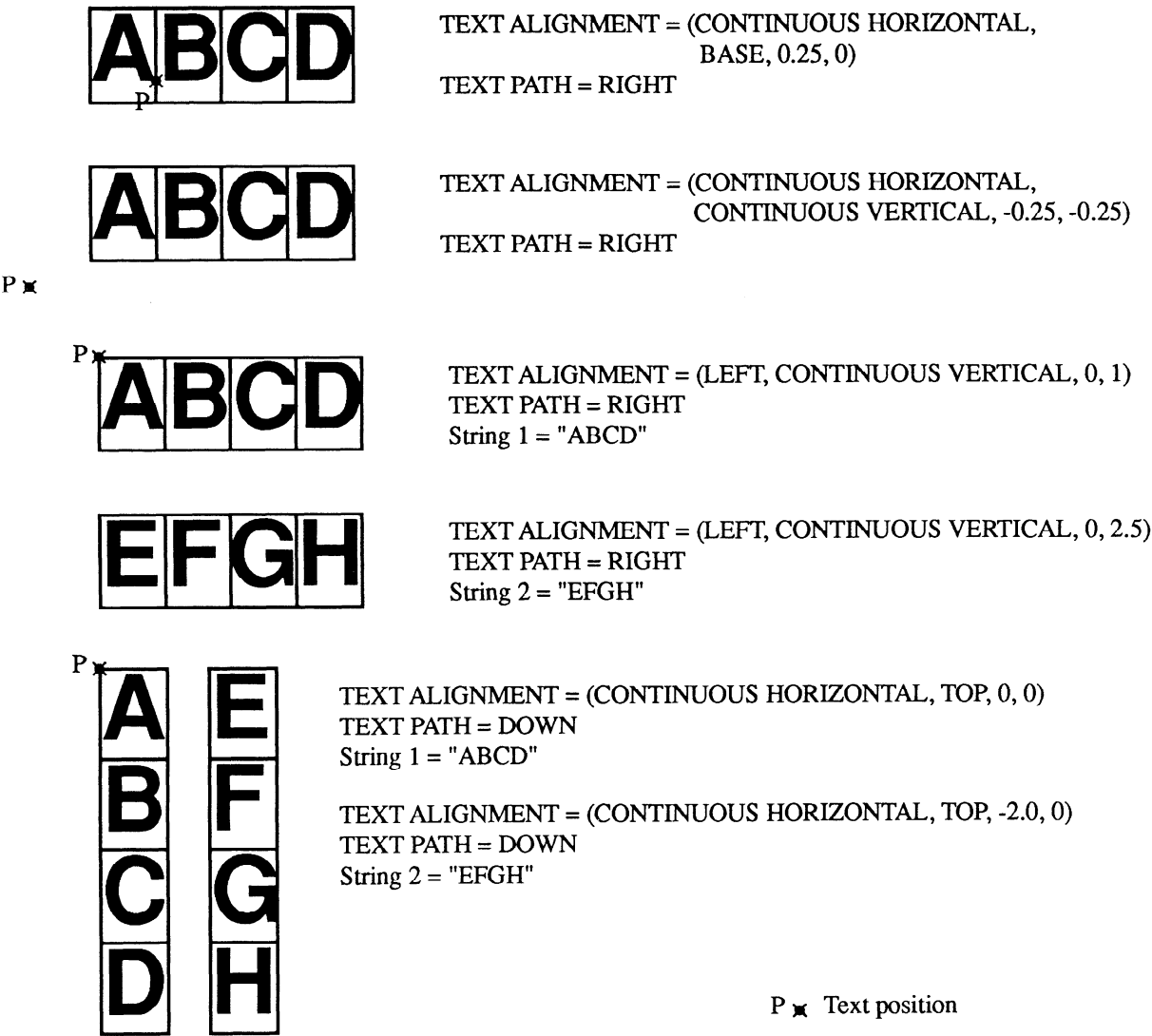


Figure 11 – Continuous text alignment

For both horizontal and vertical alignment types, normal values are converted to the appropriate value for the current text path (as indicated with the description of the TEXT ALIGNMENT function in clause 5) at the time of a text object's elaboration and thereafter treated as defined above.

Table 5 – Definition of alignment points: TEXT PATH = LEFT or RIGHT

TOP	topline farthest from the baseline
CAP	capline farthest from the baseline
HALF	halfline farthest from the baseline
BOTTOM	bottomline farthest from the baseline
LEFT	leftmost edge of leftmost character body
RIGHT	rightmost edge of rightmost character body
CENTRE	halfway between LEFT and RIGHT as defined above

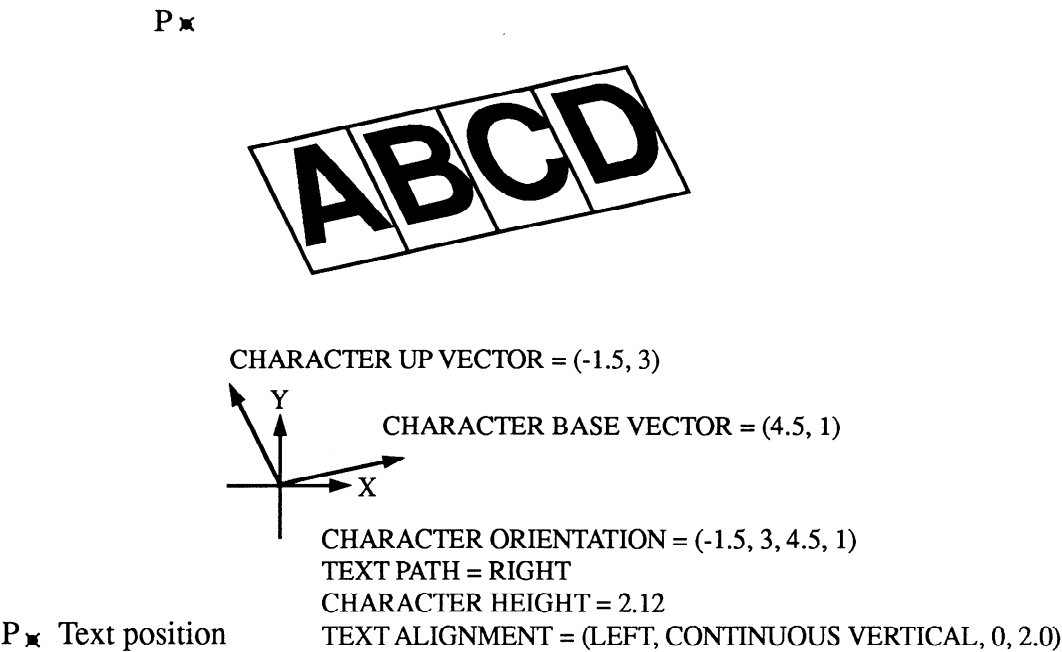


Figure 12 – Continuous text alignment with skewed orientation vectors

Table 6 – Definition of alignment points: TEXT PATH = UP or DOWN

TOP	topline of topmost character
CAP	capline of topmost character
HALF	halfway between halflines of topmost and bottommost character
BASE	baseline of bottommost character
BOTTOM	bottomline of bottommost character
LEFT	left edge farthest from the centre line
RIGHT	right edge farthest from the centre line

Note that the relationship of topline to capline, bottomline to baseline, and the placement of the halfline are font-dependent. It is for this reason that the various defining lines of the text extent parallelogram need not be derived from the same character body. This is a function of the CHARACTER HEIGHT, TEXT FONT INDEX, and CHARACTER EXPANSION FACTOR changes within a string.

The continuous values of alignment allow the display of multiline text, and may ensure that the ascenders of one row and the descenders of the next do not overlap. Use of continuous horizontal and continuous vertical alignments provides this ability for all values of TEXT PATH. Since inquiry of the dimensions of the text extent parallelogram cannot be provided in a one-way output environment, other means have been provided to align more than one row of characters to the same text position. This means that the alignment parameters are permitted to exceed the maximum dimensions of the text extent parallelogram. Use of a continuous parameter allows specification of inter-row space, analogous to intercharacter space (see figure 11).

As an example of the set of the continuous alignment attributes, consider the display of four rows of left-justified text, each specified by a single TEXT function. To ensure that ascenders and descenders do not interfere between rows and that, in addition, there is a space of at least one-half the maximum size of a character between the descenders of one row with raised accent marks or oversized symbols of another row, the TEXT ALIGNMENT function should set the horizontal and vertical alignment types to (LEFT, CONTINUOUS VERTICAL) and the continuous vertical alignment value to 0.0. Then, the first row is output with the text position equal to the lower-left corner of the string. The continuous vertical alignment value can then be set to 1.5, and the second row output with the same text position. This places the second row below the first row because of

Concepts

Text primitives

the change in alignment. The last two rows are output in the same way with the continuous vertical alignment value set to 3.0 and 4.5, and the text position parameters to the TEXT function unchanged. A value of 1.0 assures no overlap between rows; a value greater than 1.0 guarantees additional space. A similar use of CONTINUOUS HORIZONTAL alignment can be made when the text path is DOWN.

Figure 13 illustrates examples of the use of APPEND TEXT in combination with discrete text alignment attribute values.

The foregoing examples have been illustrated for the case of the character up vector and the character base vector being orthogonal (i.e. the text extent parallelogram is a rectangle). When these orientation vectors are not orthogonal, the text extent parallelogram's sides remain parallel to the two orientation vectors. The centreline is skewed to remain parallel with the left and right edges of the text extent parallelogram. The height of the text extent parallelogram is measured along the skewed edge (not perpendicular to the baseline). The TEXT PATH value RIGHT is in the direction of the character base vector and TEXT PATH value LEFT is in the opposite direction.

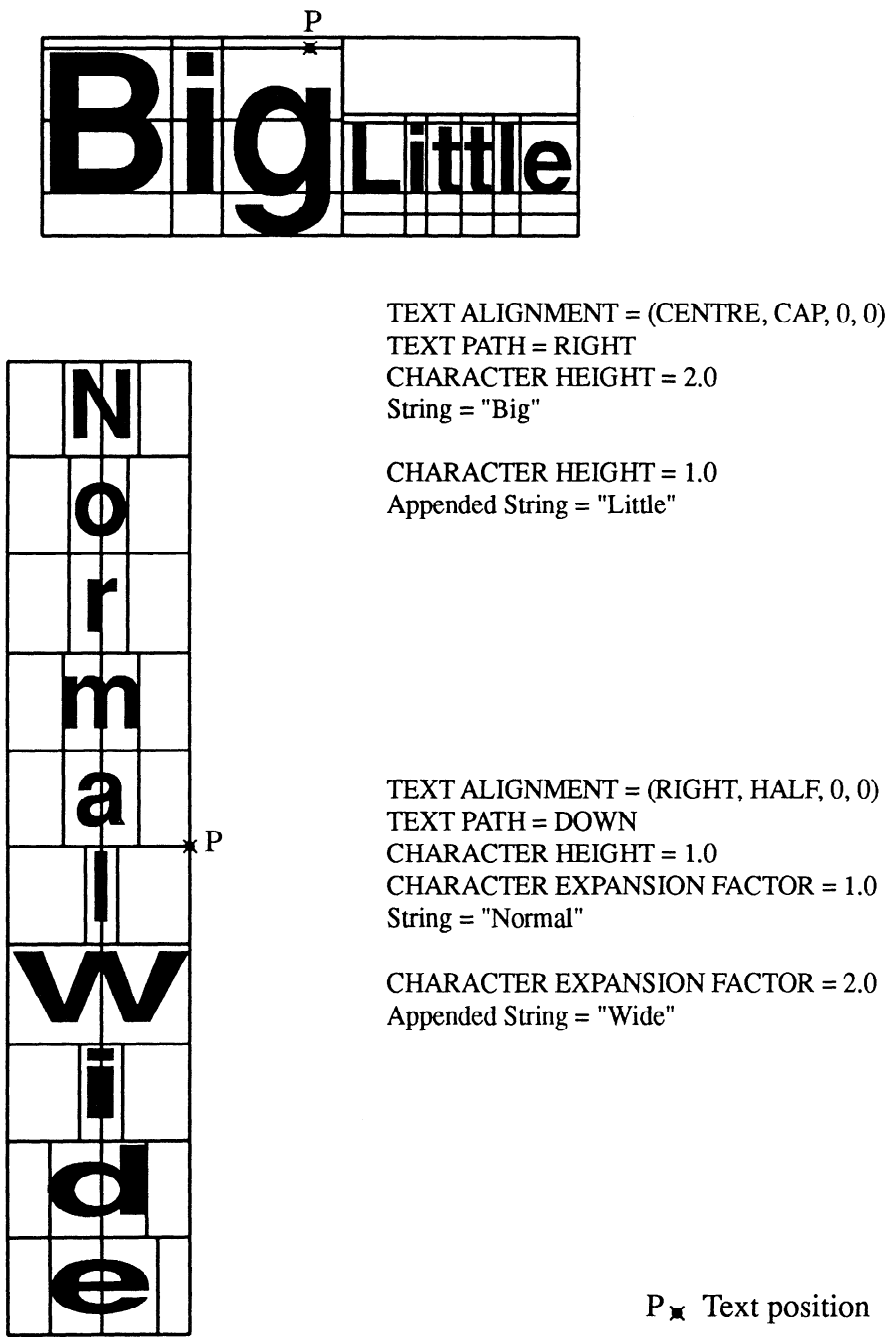


Figure 13 – Discrete text alignment with appended text and proportional spacing

3.9.4.4 Text precision

The fidelity with which the text attributes are realized on the CGI Virtual Device is controlled by the text precision attribute. Text precision may have one of the following possible values:

- STRING:** The starting position of the entire text string is evaluated correctly. The pick identifier and the text colour of the text string are also evaluated correctly. The realization of other text attributes is allowed to differ from that specified. Clipping of a text string whose text extent parallelogram intersects the boundary of the effective clip region is implementation-dependent.
- CHARACTER:** The position of each character of the rendered text string and the placement and orientation of the rendered text string satisfies the relevant text attributes, as described above. The skew, orientation, and size of each individual character is allowed to differ from that specified by the relevant text attributes. All characters of the text string which lie either entirely inside or outside the effective clip region are clipped as appropriate. Clipping of characters whose character body intersects the boundary of the effective clip region is implementation-dependent.
- STROKE:** The placement, skew, orientation, and size of all characters of the text string satisfies all the text attributes, as described above. Characters are clipped to the geometric accuracy of the Virtual Device as in SHAPE clipping.

During compound text construction, when text precision is changed explicitly or as a result of a change in the text bundle index, the highest text precision value that has been used during the definition of the compound text object is used.

3.9.4.5 Restricted text

The text attributes apply in the same way to a text object formed from the RESTRICTED TEXT function. Determination of the extent of a string is generally not possible in a 1-way output environment, hence the RESTRICTED TEXT function specifies the size of a text extent parallelogram. If the text string, as displayed by the current text attributes, would exceed the parallelogram derived from this parameter, then the text attributes CHARACTER EXPANSION FACTOR, CHARACTER SPACING, TEXT FONT INDEX, TEXT PRECISION, and CHARACTER HEIGHT are adjusted in an implementation-dependent manner so that all of the specified text string fits within the parallelogram and the extent of the displayed string does not exceed the parallelogram.

3.9.4.6 Degeneracy of text objects

Degeneracy of text objects may arise either due to the client's specifications or unintentionally due to round-off error in transforming coordinate data. However, specifying a vector of length zero for the character up vector or character base vector is prohibited. Furthermore, since only the directions and length ratio for those vectors affect rendering, an implementation should maintain a scaling of the vectors during transformation such that any problem of accidental rounding to zero (or loss of precision) is avoided.

If the Virtual Device cannot render a text object at the exact specified character height, the next smaller available implemented height will be used. If no such height is available, the next larger height available will be used.

3.9.5 Text clipping

The type of clipping that may be applied to text is dependent upon the TEXT PRECISION attribute value associated with a text graphic object, as follows:

- For STRING precision text, clipping may proceed on a per string basis, in a manner consistent with LOCUS clipping.
- For CHARACTER precision text, clipping may proceed on a per character basis, in a manner consistent with LOCUS clipping.
- For STROKE precision text, clipping always proceeds in a manner consistent with SHAPE clipping.

Note that a Virtual Device is always allowed by ISO/IEC 9636 to use SHAPE clipping for all text precisions.

3.9.6 Text fonts and character sets

3.9.6.1 Fonts in the CGI

Selection of a font (that is, the style of the characters to be displayed) from among available fonts is achieved using the TEXT FONT INDEX function. The assignment of particular font names to font indices used as values of the parameter to TEXT FONT INDEX is achieved using the function FONT LIST. The FONT LIST function may be used at various times throughout the specification of a picture. Whether changes of the List of Font Names affect the appearance of any rendered text is indicated by the Dynamic Modification Accepted For Font List entry in the Text Description Table.

The choice of character font is determined independently of the character set. However, the specified font and the character sets being used should be related in order to display meaningful text strings. Roman and Gothic are examples of commonly used fonts for Latin-based alphabets.

3.9.6.2 Using multiple character sets

There are several methods for specifying a string containing characters from different character sets. The method used is determined by the CHARACTER CODING ANNOUNCER function. The default or normal technique is to use the CHARACTER SET INDEX function, and restrict the contents of the text strings to printing characters and spaces (format effector control codes such as CR and LF are permitted, but their interpretation is implementation-dependent). Other settings of the CHARACTER CODING ANNOUNCER function or use of the ALTERNATE CHARACTER SET INDEX function permit standardized use of 8-bit characters between SI, SO, and ESC control codes within the text string, in accordance with ISO 2022. The ALTERNATE CHARACTER SET INDEX function is used to select a character set to be used as both the G1 set and the G2 set. The G1 set is used both for 8-bit characters in columns 10-15 of the code table, and with the SO control code. The assignment of particular character sets to the index parameter of both CHARACTER SET INDEX and ALTERNATE CHARACTER SET INDEX is done with the CHARACTER SET LIST function. These assignments are performed during the generation of the text primitive and are not affected by any later changes resulting from an invocation of the CHARACTER SET LIST function.

These code extension techniques apply to string parameters of text functions. For strings which are part of a data record of an ESCAPE, GET ESCAPE, or GDP function, whether or not the code extension technique applies is specified in the description of the particular escape or GDP.

3.9.7 Errors in TEXT OPEN state

3.9.7.1 Non-contributing functions

Attribute and control functions not directly contributing to the construction of compound text objects are permitted during TEXT OPEN state, and have their normal effect on the corresponding state list entries. Non-contributing primitives cause an error and are ignored.

3.9.7.2 Overflow of the text buffer

When overflow occurs during TEXT OPEN state, a single error is generated and the CGI device stays in state TEXT OPEN. Subsequent primitives which would have contributed to the compound text are ignored without generating additional errors; other behaviour and error conditions defined for TEXT OPEN state remain the same after overflow. When an APPEND TEXT with the “not final/final” flag of FINAL is invoked, as much of the compound text as had been entered into the text buffer when the overflow occurred is used and global attribute values in effect are associated with it before it is passed on down the pipeline.

3.9.8 Allowed latitude

The following levels of compound text support are permitted for a CGI Virtual Device: NONE, GLOBAL, and LOCAL. The implemented level of support is indicated by the Compound Text Capability entry in the Primitive Support Description Table.

- A level of NONE indicates that the Virtual Device does not support compound text objects in any form. That is, the functions TEXT and RESTRICTED TEXT are only permitted with the “not final/final” flag set to FINAL; APPEND TEXT is not supported.

Text primitives**Concepts**

- A level of GLOBAL indicates that the Virtual Device does not support local text attribute changes between text primitives contributing to a compound text object. All associated attribute values are taken from the state lists at the time of the “final” APPEND TEXT.
- A level of LOCAL indicates full support of compound text objects as described in 3.9.2.

3.10 Fill primitives**3.10.1 Fill functions**

The CGI provides the following functions which generate fill primitives:

POLYGON	defines an area bounded by lines defined by a list of points;
POLYGON SET	defines a set of areas bounded by lines defined by a list of points and subject to edge control flags;
RECTANGLE	defines a rectangular area oriented parallel to the VDC axes;
CIRCLE	defines a circle specified by a centre point and a radius;
CIRCULAR ARC CENTRE CLOSE	
CIRCULAR ARC THREE POINT CLOSE	defines a subregion of a circle specified by one of two parameterizations and a close type that selects between either PIE or CHORD style closure;
ELLIPSE	defines an ellipse specified by a centre point and a conjugate radius pair;
ELLIPTICAL ARC CLOSE	defines a subregion of an ellipse specified by a centre point, a conjugate radius pair, and start and end vectors, together with a closure flag that selects between either PIE or CHORD style closure.

In addition to the above fill primitives, an arbitrary filled region may be constructed as a compound object using line and fill primitives and the closed figure control functions as described in 3.10.5. Closed figure objects are filled in the same manner as other fill objects and use the same set of fill attributes.

3.10.2 Fill attributes

The appearance of fill primitives is determined by specific fill attributes, and other general attributes. (See 3.10.5 for a description of the association of fill attributes in closed figures.)

3.10.2.1 Interior attributes

INTERIOR STYLE	determines the style of filling of the interior of a fill object (see below);
FILL COLOUR	determines the colour in which filling is performed for interior styles HOLLOW, SOLID, or HATCH. The colour selection mode in which the fill colour was specified determines whether the FILL COLOUR attribute value is interpreted as a direct or indexed colour value during rendering. See 3.4 for a description of how colours are defined in the CGI;
HATCH INDEX	determines the hatch style to be used if the interior style is HATCH;
PATTERN INDEX	determines the pattern entry in the pattern table to be used if the interior style is PATTERN;
FILL BITMAP IDENTIFIER	determines the bitmap to be used if the interior style is BITMAP. The value of Fill Bitmap Identifier in the Fill Attributes State List is specified by the function FILL BITMAP defined in ISO/IEC 9636-6;
FILL BITMAP	determines the bitmap region to be used if the interior style is BITMAP. The value of Fill Bitmap Region in the Fill Attributes State List is defined by the function FILL BITMAP specified in ISO/IEC 9636-6;

Concepts

Fill primitives

FILL BUNDLE INDEX	determines the fill bundle to be used during rendering when the corresponding fill ASF values are BUNDLED. The following fill attribute values may be bundled: INTERIOR STYLE, FILL COLOUR, HATCH INDEX, PATTERN INDEX, FILL BITMAP IDENTIFIER, and FILL BITMAP REGION;
Fill ASFs	determine which attribute values of the fill bundle referenced by the FILL BUNDLE INDEX attribute are to be used during rendering when the corresponding ASF attribute value is BUNDLED. The following fill ASFs are defined: INTERIOR STYLE ASF, FILL COLOUR ASF, HATCH INDEX ASF, PATTERN INDEX ASF, and FILL BITMAP ASF. The FILL BITMAP ASF controls both fill bitmap attributes;
PATTERN SIZE	determines the spatial repeat period of the pattern during rendering if the interior style is PATTERN;
FILL REFERENCE POINT	determines the origin position of the pattern or bitmap repetition if the interior style is PATTERN or BITMAP.

The fill interior style is used to determine the style in which fill objects are to be filled. It has the following values:

HOLLOW:	the interior is not filled, but the clipped boundary of the fill object is rendered using the associated FILL COLOUR;
SOLID:	the interior is filled using the associated FILL COLOUR;
PATTERN:	the interior is filled with repetitions of the pattern selected from the pattern table by the PATTERN INDEX function; the repetitions have their origin at the FILL REFERENCE POINT attribute value and their size and shape are determined by the PATTERN SIZE attribute value;
HATCH:	the interior is filled with the hatch style selected by the associated HATCH INDEX, rendered in the associated FILL COLOUR;
EMPTY:	no representation of the interior is rendered;
BITMAP:	the interior is filled with repetitions of the bitmap region determined by the FILL BITMAP IDENTIFIER and FILL BITMAP functions. The origin of the repetitions is specified by the FILL REFERENCE POINT function.

For interior style PATTERN, the pattern is defined by a pattern table entry selected by the pattern index, which specifies an array (nx, ny) of colour specifiers. The size, shape, and start position of the pattern are specified by a pattern parallelogram, as defined by pattern width and height vectors located relative to the associated FILL REFERENCE POINT. The pattern parallelogram is conceptually divided into cells; nx cells in the width vector direction and ny cells in the height vector direction. Each cell is thus (pattern width/nx) wide and (pattern height/ny) high, as measured parallel to the sides of the pattern parallelogram.

The pattern parallelogram for interior style BITMAP, which is the Fill Bitmap Region defined through use of the FILL BITMAP function, is always rectangular and never transforms. See the FILL BITMAP function described in ISO/IEC 9636-6, 5.3.2 for specification of how the associated attributes of TRANSPARENCY and AUXILIARY COLOUR are applied in rendering this interior style.

The array of colours of the selected pattern is mapped onto the array of cells as follows: for interior style PATTERN, the colour array element (1, ny) is mapped to the pattern parallelogram cell which is located at the associated FILL REFERENCE POINT; for interior style BITMAP, pixel (1, ny) of the FILL BITMAP is located at the FILL REFERENCE POINT. Colour array elements with increasing first dimension are associated with successive cells in the direction of the width vector, and colour array elements with decreasing second dimension are associated with successive cells in the direction of the height vector. In this way, each of the nx*ny colour array elements is associated with one of the nx*ny cells of the pattern box.

Conceptually, the pattern parallelogram so defined is replicated in directions parallel to the vectors of the PATTERN SIZE until the interior of a fill object to which the pattern is to be applied is completely covered.

3.10.2.2 Edge attributes

EDGE TYPE	determines the type of edge (e.g. solid, dotted, or dashed) with which a fill object is rendered if the edge visibility is VISIBLE;
------------------	---

Fill primitives

Concepts

SPECIFICATION MODE OF EDGE WIDTH	determines whether the EDGE WIDTH attribute value is interpreted as a scaling factor or a VDC width during rendering (see 3.10.3);
EDGE WIDTH	determines the width of edge with which a fill object is rendered if the edge visibility is VISIBLE;
EDGE VISIBILITY	determines the visibility of the edges of a fill object when rendered;
EDGE COLOUR	determines the colour with which the edges of a fill object are rendered if the edge visibility is VISIBLE. The colour selection mode in which the edge colour was specified determines whether the EDGE COLOUR attribute value is interpreted as a direct or indexed colour value during rendering;
EDGE CLIPPING MODE	determines the type of object clipping to be performed during rendering (see 3.10.4);
EDGE BUNDLE INDEX	determines the edge bundle to be used during rendering when the corresponding edge ASF values are BUNDLED. The following edge attribute values may be bundled: EDGE TYPE, EDGE WIDTH, EDGE COLOUR, and EDGE VISIBILITY;
Edge ASFs	determine which attribute values of the edge bundle referenced by the EDGE BUNDLE INDEX attribute are to be used during rendering when the corresponding ASF attribute value is BUNDLED. The following edge ASFs are defined: EDGE TYPE ASF, EDGE WIDTH ASF, EDGE COLOUR ASF, and EDGE VISIBILITY ASF.

3.10.2.3 General attributes

The following general attributes are associated with fill primitives – AUXILIARY COLOUR, TRANSPARENCY, and DRAWING MODE as local attributes for edges, and global attributes for interiors; CLIP INDICATOR, CLIP RECTANGLE, and PICK IDENTIFIER as global attributes (see 3.10.5).

CLIP INDICATOR	determines whether object clipping is to be applied to the fill object;
CLIP RECTANGLE	used in determining the effective clip region to be applied to a fill object if the associated CLIP INDICATOR is ON;
AUXILIARY COLOUR	determines the colour to render the gaps in a line for edge types other than solid, or gaps between hatch lines for interior style HATCH, if the TRANSPARENCY attribute value is OPAQUE. The colour selection mode in which the auxiliary colour was specified determines whether the AUXILIARY COLOUR attribute value is interpreted as a direct or indexed colour value during rendering;
TRANSPARENCY	determines whether gaps in a broken edge or between hatch lines are rendered using the AUXILIARY COLOUR attribute value. If the TRANSPARENCY attribute value is TRANSPARENT, any previously drawn parts of the picture lying within the gaps are unaffected;
DRAWING MODE	determines how the rendered (and/or rasterized) object is mixed with what already exists on the drawing surface (defined in ISO/IEC 9636-6);
PICK IDENTIFIER	(Defined in ISO/IEC 9636-4.)

3.10.3 Fill geometry

3.10.3.1 Definition of fill object interior

For all fill objects in the CGI (including closed figures) a parity fill algorithm is used to determine those portions of the interior as follows:

- Any given point is considered inside the region if a ray cast from the given point to infinity, such that it misses all vertices and arc endpoints used to define the explicit and implicit boundary portions of the region and never intersects an arc tangentially, intersects the unclipped boundary of the region an odd number of times. Note that the ray may intersect the boundary at a single point multiple times if the boundary passes through that point multiple times.

3.10.3.2 Interaction of edge, boundary, and interior

Fill objects have a single edge which, in the case of closed figures and POLYGON SETs, may consist of several edge portions depending on the exact nature of the fill object constructed. Each edge portion has associated edge attribute values. Parts of edge portions which are clipped are discarded. Clipping of edges is performed identically to clipping of line objects, although controlled by a separate object clipping mode.

The boundary, which is rendered only in the case of interior style HOLLOW, is considered to be the representation of the interior. It is not a rendering of the edge.

If the edge portion is visible, it is rendered "in front of" the interior, i.e. the edge takes precedence over the interior when rendered. In the case of interior style HOLLOW, implicit and explicit boundary portions may be partly visible, in addition to rendered edges.

3.10.3.3 Rendering edges

Edges are rendered using edge attributes in the same way as lines are rendered using line attributes (with the exception of edge visibility). In particular, edge widths are treated in the same way as line widths (see 3.7.3).

3.10.3.4 Transformation of fill objects

Rectangles, circular, and elliptical fill objects are abstractly rendered before transformations are applied. Therefore, the entire abstract locus of these fill objects is subject to transformation. After transformation, angles in polygons may be altered: rectangles may become parallelograms, and circles may become ellipses. Transformation does not affect rendering of the interior if HOLLOW, SOLID, EMPTY, or BITMAP interior style has been selected. The implementation of the transformation of patterns is subject to allowed latitude, with the preferred behaviour that they are transformed. Hatch lines may be transformed according to the definition of the specific hatch index being used.

3.10.3.5 Degeneracy in fill objects

Degeneracy of fill objects (primitives) may arise either due to the client's specifications or unintentionally due to round-off error in transforming coordinate data.

There are a number of ways in which a region of a fill object may be reduced to one with zero area, e.g. a POLYGON where all the points are coincident or collinear. It is also possible to generate a POLYGON or POLYGON SET where a sub-region of the fill object has zero area, although the fill object, taken as a whole, is not degenerate.

Wherever such degeneracy arises, the Virtual Device shall render such a region as a dot if zero-dimensional (e.g. for a CIRCLE whose radius is or rounds off to zero) or a line if one-dimensional (e.g. for a RECTANGLE with one side of zero length but not both). This dot or line is subject to the interior attributes if edge visibility is INVISIBLE, and is subject to the edge attributes if edge visibility is VISIBLE. Note that if edge visibility is INVISIBLE and interior style is EMPTY, the degenerate region will be invisible, as would a non-degenerate fill object.

A closed circular arc 3 point or elliptical arc is non-degenerate if and only if the three specifying points yield three distinct points which are non-collinear. If the three points are all coincident, a dot is rendered. If only two distinct points are specified, a line is rendered between the points. If the three points are collinear, the arc has zero curvature; if close type is CHORD, a line is rendered from the starting point through the intermediate point to the ending point; if close type is PIE, nothing is rendered.

When such a degenerate fill primitive (or region of POLYGON SET) is invoked in FIGURE OPEN state, the implicit NEW REGIONS are performed nonetheless.

The treatment for the degeneracy of edges is analogous to the treatment for degeneracy of lines (see 3.7.3.1).

3.10.4 Fill clipping

3.10.4.1 Interior clipping

When a fill object is clipped using the associated effective clip region, the resulting new clip boundary portions become part of the fill object boundaries (see figure 14). Clip boundary portions are treated in the same way as implicit boundary portions during rendering. Note that the drawing surface clipping does not contribute new clip boundary portions.

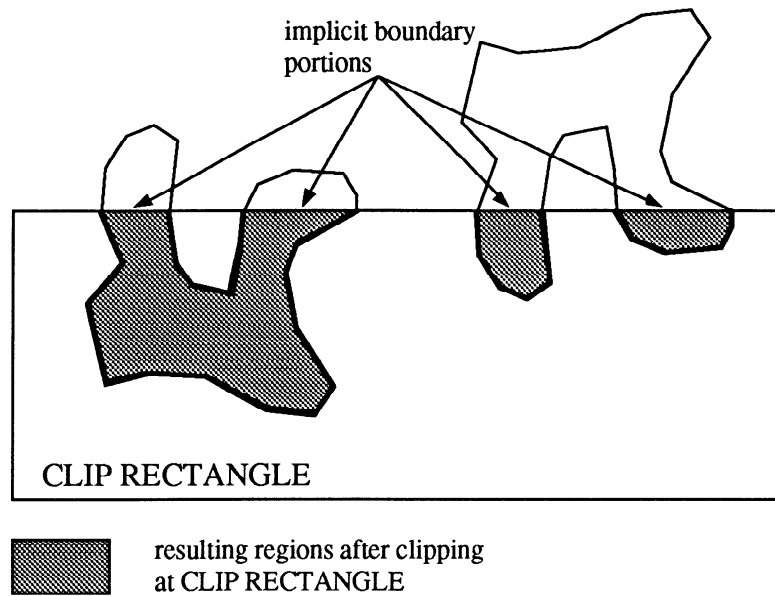


Figure 14 – Examples of fill object clipping

3.10.4.2 Edge clipping

The EDGE CLIPPING MODE attribute specifies how clipping applies to the edges of fill objects. The EDGE CLIPPING MODE attribute has the same enumeration as LINE CLIPPING MODE. Edge clipping is defined according to EDGE CLIPPING MODE analogously to line clipping (see 3.7.4).

3.10.5 Closed figures

3.10.5.1 Construction of closed figures

A closed figure is a fill type compound object which the client constructs on the device side of the interface by invoking BEGIN FIGURE, an ordered sequence of line and fill primitives (and optionally attributes and NEW REGION functions), and END FIGURE. Edge attribute values are associated locally with the edge portions and fill interior attribute values are associated globally with the complete graphic object; in addition, certain general attribute values are associated locally with edge portions and globally with the interior of the fill object. The entire fill object is considered as a single unit during subsequent processing and rendering.

3.10.5.1.1 Closure point

The first point of the first line primitive in a new region is the closure point for that region. The Virtual Device retains this closure point for use in closing the region. When the region is closed (with a NEW REGION or END FIGURE function, or by invoking a fill primitive which begins a new region) an implicit boundary portion from the last point of the last line primitive in the region to this closure point is added to the closed figure by the Virtual Device unless these points are already coincident.

3.10.5.1.2 Regions

A closed figure consists of one or more regions. A region has a closed boundary which may be concave, convex, and self intersecting. A region is formed either by invoking a fill primitive in FIGURE OPEN state (which closes the last region and contributes one or more complete regions), by invoking NEW REGION to start new regions to be formed from line primitives, or by a final invocation of END FIGURE. A closed figure constructed from only line primitives without use of NEW REGION consists of a single region.

The NEW REGION function may occur at any time during the closed figure construction. If the current region is closed, the function is ignored. If the current region is open, an implicit boundary portion is added from the last point of the last primitive to the current closure point unless CONNECTING EDGE has been invoked after the last line primitive, in which case, an explicit boundary portion and edge portion is added by the CONNECTING EDGE line primitive.

3.10.5.2 Boundaries and edges

The boundary of each region consists of a combination of explicit and implicit boundary portions. Explicit boundary portions always contribute to the edge of a region.

3.10.5.2.1 Explicit boundary portions

Explicit boundary portions and edge portions are those added by client invocation of primitives in state FIGURE OPEN. These are generated in the following situations:

- For fill primitives other than POLYGON SET, the complete edge becomes an explicit boundary portion in the closed figure.
- For POLYGON SET, the portions of the edge with the edge-out flag VISIBLE or CLOSE VISIBLE contribute explicit boundary portions.
- For line primitives, those portions which would be rendered if the state were not FIGURE OPEN become explicit boundary portions when incorporated into a closed figure.
- For DISJOINT POLYLINE in particular, only the segments from the first point to the second point, from the third point to the fourth point, and so on, become explicit boundary portions when incorporated into a closed figure.
- A CONNECTING EDGE primitive which precedes an action which would normally have added an implicit boundary portion to the closed figure either to close a region (including closing the closed figure itself) or to connect two line primitives results in the portion added being an explicit boundary portion. CONNECTING EDGE preceding or following DISJOINT POLYLINE or POLYGON SET does not affect the interpretation of those functions with respect to boundaries and edges except to introduce an explicit boundary portion connecting to the first point of a DISJOINT POLYLINE or POLYGON SET.

Explicit boundary portions and edge portions have associated edge attribute values taken from the Edge Attributes State List. These state list entries can be changed between the line and fill primitives that result in edge portions in a closed figure, and hence each edge portion has a distinct set of attribute values associated with it.

3.10.5.2.2 Implicit boundary portions

Edge attributes are never associated with implicit boundary portions. Implicit boundary portions are only rendered for interior style HOLLOW and are a special representation of the interior, not a representation of any portion of the edge.

Implicit boundary portions are added by the CGI device to the closed figure definition under the following circumstances:

- When NEW REGION, END FIGURE, or a fill primitive is invoked and the current region has not been explicitly closed and CONNECTING EDGE has not been invoked since the last line primitive: an implicit boundary portion is added from the last point of the last primitive to the current closure point to close the region.
- When the last point of the preceding line primitive is not coincident with the first point of the current line primitive, an implicit boundary portion is created to connect the last point of the preceding line primitive to the first point of the current line primitive if CONNECTING EDGE has not been invoked between the preceding and current line primitive.
- When portions of a DISJOINT POLYLINE primitive would not normally be rendered (i.e. from the second point to the third point, from the fourth point to the fifth point, and so on), implicit boundary portions are added between these points. (These are additional to the ones which may be added to connect to a preceding or following line primitive or to effect region closure after the disjoint polyline.)
- The portions of the boundary of a POLYGON SET primitive with the edge-out flag INVISIBLE or CLOSE INVISIBLE.

3.10.5.2.3 Conditions under which no boundary or edge is added

No boundary or edge portion is ever created connecting two regions, regardless of how those regions were created or closed.

3.10.5.3 Contribution of primitive functions to the closed figure**3.10.5.3.1 Contribution of line functions to the closed figure**

For line primitives, the “first point” of a line primitive is connected to the “last point” of the preceding line primitive, and the connecting implicit boundary portion becomes part of the boundary of the closed figure. For each of the CGI supported line primitives the first and last points are defined to be as follows:

Fill primitives**Concepts**

POLYLINE p1, p2, ..., pn:

p1 is the first point; pn is the last point.

DISJOINT POLYLINE p1, p2, ..., pn:

p1 is the first point; pn is the last point.

CIRCULAR ARC 3 POINT p1, p2, p3:

p1 is the first point; p3 is the last point.

CIRCULAR ARC CENTRE ...:

CIRCULAR ARC CENTRE REVERSED...:

The first point is the intersection of the circle with the ray (dx start, dy start) from the centre point (i.e. the clockwise end of the arc for **CIRCULAR ARC CENTRE**, the anti-clockwise end of the arc for **CIRCULAR ARC CENTRE REVERSED**); the last point is the intersection of the circle with the ray (dx end, dy end) from the centre point (i.e. the anti-clockwise end of the arc for **CIRCULAR ARC CENTRE**, the clockwise end of the arc for **CIRCULAR ARC CENTRE REVERSED**).

ELLIPTICAL ARC ...:

The first point is the intersection of the ellipse with the ray (dx start, dy start) from the centre point; the last point is the intersection of the ellipse with the ray (dx end, dy end) from the centre point.

GENERALIZED DRAWING PRIMITIVE...:

For GDPs which generate line primitives, the first point is the first point of the point list; and the last point is the last point of the point list, unless explicitly stated otherwise in the GDP registration or other documentation.

CONNECTING EDGE:

If the region is open, the starting point of the connecting edge is the last point of the last line primitive, and the ending point of the connecting edge is either the first point of the following primitive or the current closure point as described above. If the connecting edge would be of zero length (i.e. if the two points it connects are coincident), the function is ignored. As with other line primitives, the edge attribute values in effect at the time it is invoked are associated with any edge portion generated by this function.

If the current region is not open, invocations of the **CONNECTING EDGE** function are ignored (i.e. **CONNECTING EDGE** cannot be used to connect regions).

CONNECTING EDGE is a line primitive, not a modal setting—it should be invoked once for each connecting edge desired in the closed figure.

Invoking **CONNECTING EDGE** more than once after a line primitive results in the first instance (with its associated local attributes) being used.

The theoretical zero-width definitions of the line primitives, not their finite-width renditions on the drawing surface, are used to define the explicit boundary portions of the closed figure. In particular, clipping does not apply to the construction of the closed figure, and the gaps or spaces of the edge type or the rendered width of the edge width do not affect the definition of the boundary of the closed figure.

3.10.5.3.2 Contribution of fill functions to the closed figure

Each fill primitive contributes a complete region to the figure (**POLYGON SET** may contribute more than one), after first closing the current region if one is open. The CGI device performs an implicit **NEW REGION** before and after a fill primitive invoked in **FIGURE OPEN** state. (I.e. the new region resulting from a fill primitive is closed, and the next primitive begins a new region.)

The unclipped boundary of each fill primitive contributes to the unclipped boundary of the closed figure.

POLYGON SET primitives contribute to closed figure construction as follows:

- A **POLYGON SET** is considered to contribute one or more complete regions. If the current region has not been closed, an implicit **NEW REGION** is performed before the **POLYGON SET** is added to the figure definition. If the **POLYGON SET** does not end with a point whose edge-out flag is **CLOSE VISIBLE** or **CLOSE INVISIBLE**, an implicit **NEW REGION** is performed after the **POLYGON SET**.

- Sequences of points with edge-out flag **VISIBLE** are treated as if they were polylines, terminating with the first point with a different edge-out flag. Each such polyline becomes an edge portion of the boundary of the figure. The edge attribute values (including **EDGE VISIBILITY**) in effect when **POLYGON SET** is invoked are associated with any edge portion added in this way.
- Sequences of points with edge-out flag **INVISIBLE** contribute implicit boundary portions which are polylines joining the points in the sequence, but not edges. Edge attribute values are not associated with these.
- Points with edge-out flag **CLOSE INVISIBLE** generate the equivalent of a **NEW REGION**, generating an implicit boundary portion from this point to the current closure point if these are not coincident, and closing the current region.
- Points with edge-out flag **CLOSE VISIBLE** generate the equivalent of a **CONNECTING EDGE** followed by a **NEW REGION**, resulting in an edge portion from this point to the current closure point if these are not coincident. The edge attribute values (including **EDGE VISIBILITY**) in effect when **POLYGON SET** is invoked are associated with any edge portion added in this way.
- **POLYGON SET** does not affect the value of the Edge Visibility entry in the state list.

3.10.5.3.3 GDP

A GDP which is defined as a line primitive shall specify the first point as the first point of the point list and the last point as the last point of the point list, unless explicitly stated otherwise in the GDP registration, with respect to closed figure construction. Such GDPs are assumed to contribute to a closed figure a boundary corresponding to the unclipped locus which would be rendered if the function were invoked when not in **FIGURE OPEN** state. Any other behaviour shall be documented explicitly in the GDP description. A GDP which is defined as being a fill primitive is treated as in the previous section. Any variation or special handling in state **FIGURE OPEN** shall be documented explicitly in the GDP description.

3.10.5.4 Local and global attributes of closed figures

Local attributes are those associated with each individual edge portion, and that can vary between each edge portion within the closed figure object. The local attributes for the edge portions of closed figures are the set of edge attributes (excluding **EDGE CLIPPING MODE**) and the general attributes **AUXILIARY COLOUR**, **TRANSPARENCY** and **DRAWING MODE**.

Global attributes are those associated with the closed figure object as a whole rather than its component parts. The functions which set their state list values may be invoked during **FIGURE OPEN** state, and have their usual effect on the corresponding state list entries. The values associated with the closed figure are those in the state list when **END FIGURE** is invoked to complete the object formation (even in the event of figure buffer overflow during construction). Global attributes of closed figures are the set of fill interior attributes, the **EDGE CLIPPING MODE** attribute, the general attributes **CLIP INDICATOR**, **CLIP RECTANGLE**, and **PICK IDENTIFIER**, and the general attributes **AUXILIARY COLOUR**, **TRANSPARENCY**, and **DRAWING MODE**, which apply to the interior of the closed figure. In order to use a different value of one of the last three general attributes for the interior from that for any of the edge portions, the appropriate attribute function should be invoked just prior to invoking **END FIGURE** with the value to be used for the interior.

3.10.5.5 Errors in FIGURE OPEN state

3.10.5.5.1 Non-contributing functions

Non-contributing primitives cause an error and are ignored. Attribute and control functions not directly contributing to the construction of the closed figure object are permitted, and have their normal effect on the corresponding state list entries.

3.10.5.5.2 Overflow of the figure buffer

When overflow occurs during **FIGURE OPEN** state, a single error is generated and the CGI device stays in state **FIGURE OPEN**. Subsequent primitives which would have contributed to the closed figure are ignored without generating additional errors; other behaviour and error conditions defined for **FIGURE OPEN** state remain the same after overflow. When **END FIGURE** is invoked, as much of the closed figure as had been entered into the buffer when the overflow occurred is closed and global attribute values in effect are associated with it before it is passed on down the pipeline.

3.10.5.5.3 Abandonment of the closed figure definition

If **VDC TYPE** or any function which changes the VDC-to-Device Mapping is invoked in **FIGURE OPEN** state, the closed figure under construction is abandoned and the Virtual Device ignores subsequent primitives which would have contributed to the closed figure until the receipt of **END FIGURE** causes the transition out of **FIGURE OPEN** state. Other behaviour and error conditions defined for **FIGURE OPEN** state remain the same in this case.

Invoking INITIALIZE in FIGURE OPEN state causes the closed figure to be abandoned, the figure buffer to be cleared, and a transition out of state FIGURE OPEN.

3.10.5.6 Examples

Annex E gives some examples of the use of closed figures.

3.10.6 Allowed latitude

3.10.6.1 Interior styles

An implementation is only required to support interior style HOLLOW. The list of interior styles supported is recorded in the Fill Description Table.

When the number of pixels in the region specified by PATTERN SIZE is insufficient for displaying the selected pattern at one pattern cell per pixel, the CGI allows some latitude. The preferred behaviour is to use as much of the pattern as possible. Sampling, or otherwise condensing the pattern, is also permitted, and is indicated by the Pattern Fill Fallback entry in the Fill Description Table.

The implementation of the transformation of patterns is subject to allowed latitude, with the preferred behaviour that they are transformed.

3.10.6.2 Global and local attributes of closed figures

An implementation need not distinguish between global and local attributes in the construction of closed figure objects. The level of support provided by an implementation is indicated by the Closed Figure Capability entry in the Primitive Support Description Table as follows:

- A level of NONE indicates that the Virtual Device does not support closed figures in any form.
- A level of GLOBAL indicates that the Virtual Device does not support local edge attribute changes during the FIGURE OPEN state. That is, all local attributes are treated as global attributes of the closed figure. (I.e. the values used are those in effect at the time that END FIGURE function is invoked.)
- A level of LOCAL indicates full support of closed figures as described in 3.10.5.

3.10.6.3 Edge continuity and edge width

The following levels of Edge Type Continuity are permitted for a CGI Virtual Device: RESTART, CONTINUOUS, and OTHER. The implemented level of support is indicated by the Line/Edge Type Continuity Capability entry in the Primitive Support Description Table.

- A level of RESTART indicates that the edge type pattern is restarted at each vertex of the visible edge of a fill object except the last vertex;
- A level of CONTINUOUS indicates that the Virtual Device maintains the edge type continuously across all vertices of the visible edge of a single fill object except the first and last vertices;
- A level of OTHER indicates that another method has been used (for example, guaranteeing some continuity and that end points of each visible edge are always drawn).

ISO/IEC 9636 allows latitude in the alignment of a finite width realized edge with respect to the boundary. The preferred behaviour is to centre the line on the boundary; an allowed latitude is to render the edge everywhere inside and contiguous to the boundary.

3.11 Image primitive

3.11.1 Image function

The CGI provides a single image function which generates an image primitive:

CELL ARRAY defines a two-dimensional array of coloured cells and their position and shape in VDC space.

3.11.2 Image attributes

The appearance of an image primitive is determined by the defining colour values of the image data and the following general attributes:

3.11.2.1 General attributes

CLIP INDICATOR	determines whether object clipping is to be applied to the image object;
CLIP RECTANGLE	used in determining the effective clip region to be applied to an image object if the associated CLIP INDICATOR is ON;
DRAWING MODE	determines how the rendered (and rasterized) object is mixed with what already exists on the drawing surface (as defined in ISO/IEC 9636-6);
PICK IDENTIFIER	(defined in ISO/IEC 9636-4).

3.11.3 Image primitive geometry

A CELL ARRAY is specified by three corner points (P, Q, R), which define a parallelogram in VDC space, and a list of cell colour specifiers. P and Q delimit endpoints of a diagonal of the parallelogram, and R defines a third corner. This area is subdivided into $n_x \times n_y$ contiguous parallelograms as follows. The edge from P to R is subdivided into n_x equal intervals, and the edge from R to Q is subdivided into n_y equal intervals. The colour list consists of $n_x \times n_y$ cell colour specifiers, conceptually organized as a two-dimensional array of dimensions n_x and n_y , representing the column and row dimensions respectively. The first array element is mapped to the cell at corner P, and the last array element is mapped to the cell at corner Q. Incrementing the first index of the array corresponds to moving along the direction from P to R. Incrementing the second index of the array corresponds to moving along the direction from R to Q.

The specification of the three points P, Q, and R may result in the CELL ARRAY being rotated, mirrored, or skewed into a parallelogram.

3.11.3.1 Degeneracy of image objects

Degeneracy of image objects may arise either due to the client's specifications or unintentionally due to round-off error in transforming coordinate data.

If the parallelogram specified for a CELL ARRAY degenerates to a region of zero area (points coincident), a dot is rendered. A line is rendered if the parallelogram degenerates to a one-dimensional region, that is, the three points are collinear. The dot or line is rendered using implementation-dependent line attributes.

3.11.4 Image clipping

The clipping of an image object proceeds in a manner consistent with SHAPE clipping.

3.11.5 Allowed latitude

This part of ISO/IEC 9636 allows latitude for whether an implementation is able to fill a CELL ARRAY primitive with the contents of its array of coloured cells, and whether it is able to perform general transformations of the array of cells. The implemented levels of support are indicated by the Cell Array Fill Capability, Cell Array Alignment Capability, and Cell Array Rendering Technique entries in the Primitive Support Description Table.

The entry Cell Array Fill Capability indicates one of:

- FILLED (preferred behaviour), to mean that the implementation is able to render the array of coloured cell;
- OUTLINED, to mean that the implementation is only able to render the outline of the array of coloured cells.

The entry Cell Array Alignment Capability indicates one of:

Image primitive**Concepts**

- SKEWED (preferred behaviour), to mean that the implementation is able to transform and render a potentially non-rectangular CELL ARRAY primitive;
- AXIS ALIGNED, to mean that the implementation is only able to render a CELL ARRAY whose outline, after transformation, is aligned with the coordinate system axes. An implementation, that supports axis aligned CELL ARRAY's only ignores the point R defining the parallelogram and uses the axis aligned rectangle defined by points P and Q.

The entry Cell Array Rendering Technique indicates one of:

- EXACT (preferred behaviour), when displaying the image object on a raster device, all pixels whose centres lie beneath any part of a cell (including its mathematical boundary) are given the colour of that cell. The colour of a pixel whose centre is precisely on the boundary between two cells is implementation-dependent;
- OTHER, the image object is rendered using some other technique (for example, using dithering or Floyd-Steinberg error diffusion).

3.12 Generalized Drawing Primitives

3.12.1 GDP function

The CGI supports access to special geometric output capabilities of devices and workstations through the GENERALIZED DRAWING PRIMITIVE (GDP). By definition, a GDP function is restricted to be only those functions which contribute to the geometry of non-standard graphic objects.

A GDP includes a list of VDC points as a parameter. Zero or more of the attribute sets of the standardized graphic primitives may affect the appearance of a particular GDP. These are the sets of attributes appropriate for the specified GDP function and are selected for the GDP as part of the definition of the GDP.

Note that, if a GDP requires attributes other than those which are already defined in ISO/IEC 9636, an ESCAPE function may be used to specify these attributes. Other types of parameters can be passed to the GDP in a data record. The format of this data record is defined in the definition of the GDP.

Non-negative GDP identifiers are reserved for registration in the ISO International Register of Graphical Items; negative values are available for private use.

See 3.9.2.1 and 3.10.5.3.3 for descriptions of the use of a GDP with compound text and closed figure objects, respectively.

3.13 Inquiry

3.13.1 State lists and description tables

Output inquiry functions, as described in clause 6 of this part of ISO/IEC 9636, provide the client with the means to access the information in the output description tables and output state lists.

Details about the relationship between a description table or state list and the corresponding inquiry function(s) are described in ISO/IEC 9636-1.

3.14 Retrieval

3.14.1 Retrieval of text extent

The GET TEXT EXTENT function is provided to permit the client to obtain the concatenation point and delimiting text extent parallelogram of a text string (see figure 15). This function is particularly useful when the Virtual Device supports kerning or proportional spacing, which make it difficult for the client to predict the extent of a text string.