

Implementation of
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**

Committees responsible for this British Standard

The preparation of this British Standard was entrusted by the Information Systems Technology Standards Policy Committee (IST/-) to Technical Committee IST/31, upon which the following bodies were represented:

- Department of Trade and Industry (IT Standards Unit)
- Information Systems Committee of the Universities Funding Council
- Rutherford Appleton Laboratory

This British Standard, having been prepared under the direction of the Information Systems Technology Standards Policy Committee, was published under the authority of the Standards Board and comes into effect on 15 June 1992

© BSI 03-2000

The following BSI references relate to the work on this standard:
Committee reference IST/31
Draft for comment 90/62654 DC

ISBN 0 580 20735 8

Amendments issued since publication

Amd. No.	Date	Comments

Contents

	Page
Committees responsible	Inside front cover
National foreword	ii
Foreword	iv
Text of ISO/IEC 9636-1	1

National foreword

This British Standard reproduces verbatim ISO/IEC 9636-1:1991 and implements it as the UK national standard.

This British Standard is published under the direction of the Information Systems Technology Standards Policy Committee whose Technical Committee IST/31 has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international committee any enquiries on interpretation, or proposals for change, and keep UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

NOTE International and European Standards, as well as overseas standards, are available from BSI Sales Department, BSI, Linford Wood, Milton Keynes, MK14 6LE.

A British Standard does not purport to include all the necessary provisions of a contract. Users of British Standards are responsible for their correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, pages i and ii, the ISO/IEC title page, pages ii to iv, pages 1 to 82 and a back cover.

This standard has been updated (see copyright date) and may have had amendments incorporated. This will be indicated in the amendment table on the inside front cover.

INTERNATIONAL STANDARD

ISO/IEC
9636-1

First edition
1991-12-15

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

Part 1:

Overview, profiles, and conformance

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

Partie 1: Résumé, profils et conformité



Reference number
ISO/IEC 9636-1:1991(E)

Contents

	Page
Foreword	iv
Introduction	1
1 Scope	3
1.1 Relationship of CGI to a computing environment	4
1.2 Position of CGI in a managed environment	4
2 Normative references	4
3 Definitions and abbreviations	5
3.1 Organization of the definitions	5
3.2 Definitions	5
3.3 Abbreviations	22
4 Reference models	23
4.1 Introduction	23
4.2 Model for CGI client/target relationships	23
4.2.1 CGI configurations diagram	23
4.2.2 Types of clients	25
4.2.3 Types of targets	25
4.3 Example configurations	26
4.3.1 CGI Virtual Devices as GKS workstations	26
4.3.2 CGM interpretation using CGI	26
4.3.3 CGI Virtual Devices as 3D workstations	27
4.3.4 Input/output workstations as combinations of CGI Virtual Devices	27
4.4 Relationship to other standards	29
4.4.1 Introduction	29
4.4.2 CGI data encodings standards	29
4.4.3 CGI language bindings standards	29
4.4.4 Relation to GKS (ISO 7942:1985)	29
4.4.5 Relation to GKS-3D (ISO 8805:1989) and PHIGS (ISO/IEC 9592:1989)	29
4.4.6 Relation to CGM (ISO 8632:1987)	30
4.4.7 Relation to Procedures for Registration of Graphical Items (ISO/IEC TR 9973:1988)	30
5 Concepts	30
5.1 Introduction	30
5.2 Global CGI concepts	31
5.2.1 CGI Graphic Object Pipeline	31
5.2.2 CGI state model	37
5.2.3 Description tables	37
5.2.4 State lists	37
5.2.5 Static and dynamic state lists	38
5.2.6 The structure of description tables and state lists	38
5.2.7 The principles of inquiry	38
5.2.8 Error philosophy	39
5.2.9 Method of describing functions	41
5.2.10 Data types employed	41
5.2.11 Formal grammar	44
5.3 Overview of ISO/IEC 9636-2: Control	44
5.4 Overview of ISO/IEC 9636-3: Output	44
5.5 Overview of ISO/IEC 9636-4: Segments	45
5.6 Overview of ISO/IEC 9636-5: Input and echoing	45
5.7 Overview of ISO/IEC 9636-6: Raster	45
6 Profiles	45
6.1 Profile definition	46

	Page
6.1.1 Constituency Profile registration	47
6.2 Foundation Profiles	47
6.3 Constituency Profiles	48
6.3.1 GKS Constituency Profiles	48
6.3.2 CGM Constituency Profile	48
6.3.3 Advanced 1-Way Output Constituency Profile	48
6.4 Foundation Profile definition	48
6.4.1 2-WAY OUTPUT Foundation Profile	48
6.4.2 1-WAY OUTPUT Foundation Profile	49
6.4.3 <input class> INPUT Foundation Profile	49
6.5 GKS Profile definition	50
6.5.1 GKS OUTPUT-0 Profile	50
6.5.2 GKS OUTPUT-1 Profile	53
6.5.3 <input class> GKS INPUT-b Profile	56
6.5.4 <input class> GKS INPUT-c Profile	57
6.5.5 GKS OUTIN-0b Profile	59
6.5.6 GKS OUTIN-0c Profile	59
6.5.7 GKS OUTIN-1b Profile	59
6.5.8 GKS OUTIN-1c Profile	59
6.6 CGM Profile definition	59
6.6.1 BASIC CGM Profile	59
6.7 Other Constituency Profile definitions	61
6.7.1 ADVANCED 1-WAY OUTPUT Profile	61
7 Classification and designation	63
7.1 Implementation conformance	63
7.2 Client conformance	63
7.3 Extensions	64
7.4 Inquiry	64
7.5 Parsing	64
Annex A (normative) Function identifiers	65
Annex B (normative) Profile identifiers	73
Annex C (informative) Guidelines for bindings and encodings	73
Annex D (informative) Format guidelines for definition of Constituency Profiles	79
Annex E (informative) CGI in a distributed system model	80
Annex F (informative) Typographic conventions	82
Figure 1 — CGI Client/Target Configurations	24
Figure 2 — CGI Virtual Devices as GKS workstations	26
Figure 3 — CGM interpretation using CGI	27
Figure 4 — CGI Virtual Devices as 3D workstations	28
Figure 5 — Implementing input/output workstations as combinations of CGI Virtual Devices	28
Figure 6 — CGI Graphic Object Pipeline	32
Figure 7 — Additional raster pipeline components	36
Figure 8 — Distributed system example	81
Table 1 — Parts of the CGI Functional Specification	3
Table 2 — GKS Constituency Profiles	48
Table 3 — Foundation Profile Identifiers	73
Table 4 — Constituency Profile Identifiers	73

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 9636-1 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.*

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

Introduction

Purpose

The Computer Graphics Interface (CGI) specifies a standard interface between device-independent and device-dependent parts of a graphics system. ISO/IEC 9636 specifies sets of functions for control and data exchange over this interface. This interface may be implemented as a software-to-software interface (as a data stream encoding used in a network environment or as a procedural binding to one or more software packages), or as a software-to-hardware interface (as a data stream encoding to one or more devices presenting a standardized Computer Graphics Interface as their device protocol). Related standards specify data stream encodings (for use in the software-to-software case, over a network, and in the software-to-hardware case), and procedural bindings (for use in the software to software case).

ISO/IEC 9636 includes a reference model and a description of the CGI's relationship to other standards.

ISO/IEC 9636 only defines graphics functions, graphics control functions, and functions to control data representations and deferral in data stream encodings. ISO/IEC 9636 does not define the protocols to be used to convey these functions between the parts of a (potentially distributed) system.

Benefits

- | | |
|-------------|---|
| Intrinsic | The CGI will simplify the development and implementation of graphics systems. ISO/IEC 9636 will encourage a uniform access to the graphics devices within an installation. As new devices are made available, graphics device drivers adhering to this interface can be installed for use by existing programs. |
| Interchange | ISO/IEC 9636 promotes the exchange of software between installations. By isolating the device-dependent aspects of any graphics system, modularity is encouraged, which promotes increased portability. The standard set of functions, access mechanisms, and terminology will allow developers and users to move between installations with minimal retraining. |
| Educational | The standard set of functions uses a standard terminology. This allows both the academic and industrial communities to develop instructional programs concentrating on programming techniques and methodologies based on these standard functions. |
| Economic | <p>In view of the trend towards lower hardware and higher software costs, the following benefits accrue from ISO/IEC 9636:</p> <ul style="list-style-type: none"> — It encourages transporting of software between installations, thereby reducing costs associated with “reinvention”; — It protects the large software investment made by both users and vendors because the software will not be rendered obsolete by the introduction of new devices; — It allows developers of new software to focus on higher-level graphics functions and applications instead of device-level functions; — It reduces maintenance of software systems because the standard encourages modularity; — It increases vendor independence for the user because any system designed to use a particular device can more easily be changed to use some other device; — It allows vendors to develop and market devices that will easily interface to the customer's system; — It enables users, manufacturers and vendors to take advantage of new, lower-cost graphics hardware designs. The total system's hardware cost may be reduced because system redesign may not be necessary. |

Design requirements

To realize the benefits described above, a number of design principles have been adopted:

- a) The Computer Graphics Interface should provide a suitable set of functions for the description of a wide range of pictorial information;
- b) The Computer Graphics Interface should provide a suitable set of functions for the necessary CGI session control of a wide range of graphics devices;
- c) The Computer Graphics Interface should address the more usual and essential features found on graphical devices directly and should provide access to less common facilities;

- d) The design of the Computer Graphics Interface should not preclude extension of ISO/IEC 9636 at a later stage to cover facilities currently not standardized;
- e) The Computer Graphics Interface should be usable from GKS (Graphical Kernel System — ISO 7942). In particular, the CGI should include functional capability to support the various levels of a GKS workstation in an efficient and concise manner, without compromising the ability of the interface to support non-GKS systems in an efficient and concise manner;
- f) The Computer Graphics Interface should be compatible with the Computer Graphics Metafile — ISO 8632. In particular, those CGM elements not associated with the file-oriented aspects of the CGM shall have corresponding CGI functions which have identical abstract names and parameterization;
- g) ISO/IEC 9636 should address the needs of different applications that have conflicting requirements for
 - allocation of processing burden between host and device;
 - speed of generation and interpretation of functions;
 - ease of transfer through different transport mechanisms.

Design criteria

The above requirements were used to formulate the following design criteria:

- a) Completeness
In any area of ISO/IEC 9636, the functionality specified by ISO/IEC 9636 should be complete in itself.
- b) Conciseness
Redundant functions or parameters should be avoided.
- c) Consistency
Contradictory functions should be avoided.
- d) Extensibility
The ability to add new functions and generality to ISO/IEC 9636 should not be precluded.
- e) Fidelity
The results and behaviour of functions should be well defined.
- f) Implementability
A function should be able to be efficiently supported on most host systems and or graphics hardware.
- g) Orthogonality
Independent functions for separate and noninteracting activities should be provided.
- h) Predictability
The recommended or proper use of a standard function should guarantee the results of using that particular function.
- i) Standard practice
Only those functions that reflect existing practice, that are necessary to support existing practice, or that are necessary to support standards being developed concurrently should be standardized.
- j) Usefulness
Functions should be powerful enough to perform useful tasks.
- k) Well-structured
The number of assumptions that functions make about each other should be minimized. A function should have a well-defined interface and a simply stated unconditional purpose. Multi-purpose functions and side effects should be avoided.

Parts of the CGI functional specification

ISO/IEC 9636, the functional specification of the Computer Graphics Interface, consists of a number of parts presenting portions of the CGI functionality, including an overview in this part of ISO/IEC 9636.

Table 1 — Parts of the CGI Functional Specification

Part No.	Title
ISO/IEC 9636-1	Overview, profiles, and conformance
ISO/IEC 9636-2	Control
ISO/IEC 9636-3	Output
ISO/IEC 9636-4	Segments
ISO/IEC 9636-5	Input and echoing
ISO/IEC 9636-6	Raster

This part of ISO/IEC 9636 gives a general overview and introduction to the basic concepts and principles of ISO/IEC 9636. It includes a reference model, the relationship to other standards, and profiles. In addition, it contains overviews of each of the subsequent parts. This part of ISO/IEC 9636 thus establishes the framework for all the parts of ISO/IEC 9636; it does not contain functional descriptions.

The functional capability provided by the CGI is separate from the specification of any particular encoding format or language binding.

1 Scope

ISO/IEC 9636 establishes the conceptual model, functional capability, and minimum conformance requirements of the Computer Graphics Interface (CGI). It specifies design requirements for encodings of the CGI. ISO/IEC 9636 defines a set of CGI functions that is expected to satisfy the following needs of a majority of the computer graphics community:

- provide an interface standard for computer graphics software package implementors;
- provide an interface standard for computer graphics device manufacturers and suppliers;
- provide an inquiry and response mechanism for graphics device capabilities, characteristics, and states;
- provide a standard graphics escape mechanism to access non-standard graphics device capabilities;
- allow for future functional extension of the CGI.

In addition to the CGI functionality, device classes, and Foundation and Constituency Profiles are defined. The device classes included in the CGI are output (OUTPUT), input (INPUT), and output/input (OUTIN). Profiles allow subsets of the CGI functions and features to be defined to suit particular well identified groups of users. There is also provision for Constituency Profiles to be registered after ISO/IEC 9636 is published. The Computer Graphics Interface (CGI) is a standard functional and syntactical specification of the control and data exchange between device-independent graphics software and an implementation of a CGI Virtual Device.

The syntax of the CGI, presented in ISO/IEC 9636, is an encoding-independent and binding-independent specification. Any similarity of the examples or function specifications to a particular encoding technique or language is coincidental unless explicitly stated otherwise.

The functions specified provide for the representation of a wide range of two-dimensional pictures and for control over their display on a wide range of graphics devices. The functions are split into groups that perform device and CGI session control, specify the data representations used, control the display of the picture, perform basic drawing actions, control the attributes of the basic drawing actions, acquire data from input devices, and provide access to non-standard device capabilities.

This part of ISO/IEC 9636 gives an overview of ISO/IEC 9636, explains the relationship between its parts and their relation to other standards, describes a reference model for graphics systems, and defines certain Foundation and Constituency Profiles. ISO/IEC 9636-2, ISO/IEC 9636-3, ISO/IEC 9636-4, ISO/IEC 9636-5, and ISO/IEC 9636-6 specify the CGI functions for different functional areas using an abstract notation.

ISO/IEC 9637 and ISO/IEC 9638 define standard data stream encodings, procedural library bindings, and single entry point procedural bindings of the CGI.

1.1 Relationship of CGI to a computing environment

ISO/IEC 9636 describes graphical services provided by a Virtual Graphics Device. The model for description of these services is expressed in terms of graphical capabilities of a single instance of a hypothetical graphics device. In all but the simplest of computing environments, CGI functions alone will not be sufficient to provide complete control over a device. Additional functions, not included in ISO/IEC 9636, will likely be needed. Examples of such functions include

- means to configure (sets of) physical devices to be accessed as CGI Virtual Devices;
- means to control a device capable of offering CGI-defined services as well as other, non-CGI-defined services, such as those implied by ISO 2022 and ISO 6429;
- means to differentiate among separate instances of CGI Virtual Devices in the same computing environment;
- means of defining or determining communication paths from CGI clients to CGI Virtual Devices.

In some cases, other standards exist that describe the functions required. For example, various communications standards address the needs of the last point above. In other cases, no standards may exist, but the tasks indicated are outside the scope of ISO/IEC 9636.

1.2 Position of CGI in a managed environment

There exists a large and growing family of computer controlled display systems that have the ability to act as if they are multiple individual display devices. Resources, most notably the visible drawing surface resources, are coordinated by the display system so that multiple non-cooperating client programs can each access the services of a separate individual device while all are actually running in a single managed environment.

The graphical capabilities of the CGI Virtual Device may suffice, in some instances, as the basis for implementing a complex, multiple-client display system. However, the complete needs of such a system are quite complex, include many non-graphical services, and (as current practice shows) are quite technology dependent. The CGI does not, therefore, purport to be a generally sufficient interface on which a managed display environment may be built. Rather, within a managed environment, the CGI will be one of the managed interfaces in a way not visible to the CGI client without recourse to services not part of ISO/IEC 9636. The use of the CGI as a managed interface within a managed display environment is not limited to raster devices.

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 2382-13:1984, *Data processing — Vocabulary — Part 13: Computer graphics*.

ISO 6429:1988, *Information processing — Control functions for 7-bit and 8-bit coded character sets*.

ISO 7942:1985, *Information processing systems — Computer graphics — Graphical Kernel System (GKS) functional description*.

ISO 7498:1984, *Information processing systems — Open systems interconnection — Basic reference model*.

ISO 8632-1:1987, *Information processing systems — Computer graphics — Metafile for the storage and transfer of picture description information — Part 1: Functional specification*.

ISO 8632-2:1987, *Information processing systems — Computer graphics — Metafile for the storage and transfer of picture description information — Part 2: Character encoding*.

ISO 8632-3:1987, *Information processing systems — Computer graphics — Metafile for the storage and transfer of picture description information — Part 3: Binary encoding*.

ISO 8805:1988, *Information processing systems — Computer graphics — Graphical Kernel System for Three Dimensions (GKS-3D) functional description*.

ISO 9282-1:1988, *Information processing — Coded representation of pictures — Part 1: Encoding principles for picture representation in a 7-bit or 8-bit environment*.

ISO/IEC 9592-1:1989, *Information processing systems — Computer graphics — Programmer's Hierarchical Interactive Graphics System (PHIGS) — Part 1: Functional description*.

ISO/IEC 9636-2:1991, *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 2: Control*.

ISO/IEC 9636-3:1991, *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 3: Output*.

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:—, *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding — Part 1: Character encoding¹⁾*.

ISO/IEC 9637-2:—, *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding — Part 2: Binary encoding¹⁾*.

ISO/IEC TR 9973:1988, *Information processing — Procedures for registration of graphical items*.

ANSI/IEEE 754 — *Standard for Binary Floating Point Arithmetic*.

3 Definitions and abbreviations

3.1 Organization of the definitions

The following list of definitions describes the usage of words or phrases which have been given special technical meaning in the context of ISO/IEC 9636. These meanings are not in conflict with dictionary meanings or usage in other ISO and IEC standards. However, there are usually additional semantics associated with the words or phrases which are essential to understanding ISO/IEC 9636.

The list of definitions is arranged in alphabetical order of main entry. Occasionally a term will be used with or without optional qualification in the text; in such cases, the optional qualification appears in parentheses in the term. Under each main entry, certain derived terms may also be covered; all such derived terms are underlined. With a few exceptions, where the appropriate main heading may not be immediately obvious, derived terms do not have separate main entries. Where a phrase appears in italic text, its definition may be as another main entry or it may be a derived term — in which case it will be found either under an entry for one of its constituents (not necessarily its first), under another compound entry with the same first constituent, or under an entry for another word with the same root.

3.2 Definitions

For the purposes of ISO/IEC 9636, the following definitions apply.

3.2.1

acknowledgement

an action performed automatically by a *logical input device* signals to the operator that an *input operation* has been completed. Initially enabled by default, this action is controllable by the *client* using acknowledgement control. In a *remote echoing* situation, the acknowledgement has to be explicitly performed by the client. Various acknowledgement may be available and the client can select from these different styles of signalling

3.2.2

active (for a LID)

a *logical input device* is active if events or sampling have been enabled or if request or echo request input is in progress; otherwise, it is not active

¹⁾ To be published.

3.2.3

allowed latitude

there are a number of cases in which ISO/IEC 9636 identifies a limited set of allowed behaviours for functions or features and allows implementors latitude to provide precisely one of this set. In such cases, a *description table* entry gives a *client* the opportunity to inquire the implemented behaviour. ISO/IEC 9636 always identifies one of these allowed behaviours as a preferred behaviour. This method of regulating implementation variability contrasts with that referred to as *implementation dependence*. Where ISO/IEC 9636 allows latitude in the value of a state list default, then a particular value is identified as the preferred default. An example is the default VDC Type, which has allowed latitude in its value, and a preferred default of INTEGER

3.2.4

appending (to a segment)

see under *segment definition*

3.2.5

arming (a trigger)

an action which enables a *trigger* to be *fired* by the *operator* to signal the occurrence of something significant. A trigger cannot be fired unless it has been armed. Triggers are armed for a *logical input device* while events are enabled for it and while a request or echo request input operation is in progress

3.2.6

aspect ratio

the ratio, using a uniform metric, of the width to the height (i.e. x to y) of a rectangular area, such as a *VDC extent* or a *device viewport*

3.2.7

aspect source flag (ASF)

aspect source flags which are themselves *attribute values* indicate the source of other attribute values which determine *aspects* of *primitives* or *objects*. If the value of a particular aspect source flag is INDIVIDUAL, then the attribute value for that particular aspect is *associated* with the primitive or object. If the value of a particular aspect source flag is BUNDLED, then the corresponding *bundle index* attribute value associated with the primitive or object is used to access a *bundle table* to find the attribute value determining the aspect in question

3.2.8

aspects (of primitive and object)

characteristics determining the visual (rendered) appearance of *primitives* or *objects* are known as aspects. A particular aspect of a particular primitive or object is determined by an *attribute value* which may be directly *associated* with it or indirectly associated with it by means of a *bundle index* as determined by the appropriate *aspect source flag*. There are other characteristics of primitives and objects which are determined by attribute values, but if these are not concerned with visual appearance or are only specifiable directly (i.e. they cannot be bundled), they are not referred to as aspects

3.2.9

association (of attribute values)

this refers to various steps in the *graphic object pipeline* where the connection between *graphic objects* and *attribute values* relevant to their *rendering* is established. For *aspects* relevant to a given *primitive type*, the corresponding attribute values are associated during the initial step of object creation and later, just before rendering, for those aspects for which the (already associated) *aspect source flag* values are bundled. For objects stored in segments, the associated object transformation attribute value can be modified by the concatenation of a segment transformation, which is a segment attribute, or a copy transformation, which is a parameter to the copy segment function. Associated attribute values can be replaced by means of the copy segment function and the inheritance filter

3.2.10

association (of triggers with LIDs)

see under *trigger association*

3.2.11**attribute function**

a function which sets an *attribute value* in a *state list*. unqualified, the term is used for functions which set *primitive* or *object* attribute values. If other attributes are intended, the term will be appropriately qualified, e.g. segment attribute function or raster attribute function

3.2.12**attribute value**

certain characteristics of entities are determined by attribute values. When used without qualification, the term implies *primitive* or *object* attribute value. All other uses of attribute value will be appropriately qualified such as *segment attribute value*, *raster attribute value*, etc. Many primitive or object attribute values are directly concerned with rendered appearance and determine *aspects*. Others may not be directly or at all concerned with rendered appearance (such as *ASFs* and pick identifiers). When a single primitive forms a single object, there is no distinction between primitive attribute value and object attribute value. However, when several primitives contribute to one object (*compound text* or *closed figure*), the primitive attribute values are associated with each contributing primitive (see *local attribute value*) while object attribute values are associated with the object as a whole (see *global attribute value*). As an example, auxiliary colour is a primitive (local) attribute value for *edges* but an object (global) attribute value for *interiors* of *fill objects*

3.2.13**auxiliary colour**

the auxiliary colour attribute value *associated* with an *object* is used in conjunction with the transparency attribute value. See under *transparency*

3.2.14**background colour**

the colour used in preparing the *drawing surface*

3.2.15**background colour (of mapped bitmap)**

a *control value* that determines the full depth value to which the background *pixels* of a *mapped bitmap* are expanded when they participate in *raster operation functions*

3.2.16**binding**

a binding is a (concrete) realization of an abstract functional specification in a programming language. This may be in terms of a procedural library binding to a standard programming language with approximately one call per function or a procedural single entry point binding of the abstract functions with all input and output parameters. Procedural single entry point bindings are based on data stream encodings; see *encoding*. In the absence of any clear context implying the contrary, “procedural binding” or “binding” may be taken to refer to a procedural library binding

3.2.17**bitblt**

BIT aligned BLock Transfer. The transfer or combination of the *pixel values* in rectangular *bitmap regions* of *bitmaps*. The CGI provides two and three operand bitblt functions where there are one or two source bitmap regions respectively, and a destination bitmap region. Arbitrary logical combinations and some arithmetic combinations are supported by *drawing mode*. See *raster operation functions*

3.2.18**bitmap**

a resource of the *Virtual Device* which may be viewed by the *client* as a rectangular array of *pixels*. This resource is available only on devices offering the functional capability defined in ISO/IEC 9636-6. Some bitmaps are inherent resources of the device; others can be defined by the client. CGI provides for full depth bitmaps, where the values assignable to the individual pixels span the total colour or grey scale capability of the device, and for mapped bitmap, where the values assignable to a pixel are only “foreground” or “background”; it does not provide arbitrary variable depth bitmap capability. Any bitmap may be selected as the drawing bitmap which is the destination (*drawing surface*) for the *rendering* of (graphic) *objects* as well as for *raster operation functions*. Bitmaps which are displayable bitmaps may also be selected as the display bitmap

3.2.19**bitmap region**

see *region*

3.2.20**blocked (of a queue)**

a queue is said to be blocked if the *client* has taken specific action to prevent the entry of elements into it. Contrast with *overflow* where there may or may not be implicit system action to “block” a queue in overflow state

3.2.21**boundary**

the boundary of a *fill object* consists of explicit boundary portions, implicit boundary portions, and clip boundary portions. The boundary of a fill *primitive* consists of the mathematical locus of its defining perimeter. During *closed figure* construction, the mathematical locus of each line primitive used (including CONNECTING EDGE) and the boundary of each fill primitive used defines an explicit boundary portion. Implicit boundary portions are those added automatically during closed figure construction to ensure closure. (The interpretation of the significance of edge out flags in the POLYGON SET function means that it can also introduce both explicit boundary portions and implicit boundary portions.) When a fill object undergoes locus clipping, pieces of the original closed boundary lying outside the effective clip region are discarded and clip boundary portions are added to maintain closure. Drawing surface clipping does not contribute any clip boundary portions. Clip and implicit boundary portions have no associated *attribute values*. Edge portions correspond to the explicit boundary portions and are rendered in accordance with their associated edge attribute values. Contrast with *edge*, and see also *interior*, *realized edge*, and *realized interior*

3.2.22**break action**

a specific action by the *operator* to indicate that an *input operation* should be aborted. Contrast with *firing* (*a trigger*) and with *timeout*

3.2.23**bundle**

virtual devices have a conceptual resource known as bundle tables. Bundle tables contain *attribute values* determining *aspects* of *primitives* or *objects*. There are different bundle types — one each for line, marker, text, (fill object) edge, and fill (object interior) aspects. A bundle table consists of all bundles of a given type; each of which is uniquely identified by its index. For a particular primitive or object, those aspects for which an attribute value from a bundle table is to be used must have the appropriate aspect source flag set to BUNDLED. In this case, the bundle which contains the relevant values is specified by another attribute value, the bundle index into the table of the appropriate type

3.2.24**cell (of CELL ARRAY or pattern)**

a subspace of *VDC space* may be divided into a regular array of smaller spaces referred to as cells. Cells occur in CELL ARRAY and in *patterns* used for interior filling of *fill objects*. In general, the subspace and the individual cells may be rectangular or skewed (i.e. a parallelogram) and of arbitrary *aspect ratio*. Some devices may not provide full support for arbitrary sized or skewed cells

3.2.25**cell (of character cell)**

the subspace of *VDC space* determined by text *attribute values* (and the positioning information for the containing text *object*) which is conceptually “occupied” by a character. This is a parallelogram; but since attributes can change within a text object and proportional spacing and kerning may be relevant, character cells may not be a uniform array as in the above definition. A character cell is relevant for positioning of subsequent character cells and also for where *auxiliary colour* is *rendered* in the remainder of the cell after the rendering of the actual glyph in *foreground colour* (in this case, text colour)

3.2.26**CGI Generator**

any agent which forms an *encoding* or *binding* of CGI functions and passes this to a *CGI interpreter*. The same agent may also interpret any responses

3.2.27**CGI Interpreter**

an implementation of a *CGI Virtual Device* which receives an *encoding* or *binding* of the CGI functions, performs the appropriate actions, and provides any required responses

3.2.28**character set**

a set of displayable glyphs represented by the individual values occurring in a text *primitive*. The particular values in the strings are dependent on the character set or the alternate character set which is referenced by the character set index *attribute value* of the text primitive and are subject to character coding announcer control. Character set is concerned with (logical) alphabet and special glyphs (national character sets, Katakana, etc.) and is orthogonal to *font* which determines the typeface and style of the actual *rendering* of the glyphs (e.g. italic, bold)

3.2.29**class**

see *input class* and *device class*

3.2.30**client**

any agent invoking functions of the abstract functional capability defined in ISO/IEC 9636; i.e. any entity invoking a *procedural binding* or generating an output data stream *encoding*. Contrast with *operator* and *CGI interpreter*

3.2.31**clip boundary portion**

see *boundary*

3.2.32**clipping**

a process by which portions of (graphic) *objects* which fall outside an *associated* object clip rectangle or portions of a *rendering* of a picture or raster operation which fall outside a *drawing surface* clip rectangle are discarded. Whether or not such discarding takes place is determined by the object or drawing surface clip indicator. The object clip rectangle and indicator are object *attribute values* whereas the drawing surface clip rectangle and indicator are *control values*. See *object clipping mode*

3.2.33**closed figure**

a *fill object* constructed on the device side of the CGI interface from a series of interface functions bounded by a BEGIN FIGURE and an END FIGURE. A closed figure consists of one or more *regions*. The *boundary* of the closed figure and of each region is constructed from explicit line, fill, and GDP *primitives* and *implicit boundary portions* added as necessary to ensure the boundary remains closed. Edge *attribute values* are *associated* with individual primitives used to construct the closed figure; these are *local attributes values*. (Interior) fill attribute values are associated with the fill object as a whole; these are *global attributes values*

3.2.34**closure point**

in *closed figure* and POLYGON SET construction, when the definition of the *boundary* of a *region* has been started but is not yet complete, the first point specified for the boundary of the region is referred to as the current closure point. The occurrence of an edge flag of CLOSE VISIBLE or CLOSE INVISIBLE in POLYGON SET or the invocation of a fill primitive, NEW REGION, or END FIGURE function while in state FIGURE OPEN (for closed figures) causes the automatic creation of a line segment constituting a portion of the boundary from the last line primitive end point to the current closure point. This line segment is either an *implicit boundary portion* or an edge portion, depending on whether CONNECTING EDGE precedes the closure

3.2.35**colour selection mode**

a mode determining whether colour specification is DIRECT by means of a (*direct*) *colour value* or INDEXED by means of an index value into a *colour table*. The colour values in both DIRECT colour specification and appearing as entries in the colour table are specified using an RGB additive colour model. Colour selection mode is independent of the setting of any colour ASFs

3.2.36**colour table**

a table of (*direct*) *colour values* used to map from an index value to a direct colour value when the *colour selection mode* is INDEXED

3.2.37**(direct) colour value**

a 3-tuple specifying the relative amounts of red, green, and blue light which must be combined to achieve the desired colour stimulus. See *colour selection mode*

3.2.38**compound object**

a generic term for both *compound text* and *closed figures*

3.2.39**compound text**

an *object* constructed on the device side of the CGI interface from a series of interface functions bounded by a TEXT or RESTRICTED TEXT with a not-final/final flag value of NOT FINAL and an APPEND TEXT with a flag value of FINAL. (Such construction might also be started, continued, or completed by GDPs.) The text attributes are subdivided into those which are *local attribute values associated* with the individual component *primitives*, and those which are *global attribute values* associated with the compound text as a whole

3.2.40**conjugate radius pair (CRP)**

a pair of radii of an ellipse such that a tangent to the ellipse at each endpoint is parallel to the other radius

3.2.41**control functions**

a function which requests some control action by the *Virtual Device*. Unqualified, this term refers to the general device and coordinate space control defined in ISO/IEC 9636-2 or the *graphic object pipeline* controls defined in ISO/IEC 9636-3. For other controls, such as raster and input, the term will always be appropriately qualified

3.2.42**control value**

a *modal* value which affects the operation of processes in the *graphic object pipeline* which are *downstream* of the *segment storage*. Contrast with *attribute value*. (Object) attribute values and control values are disjoint sets. Note, however, that *segment attribute* values which do not affect the operation of processes downstream are actually control values in the sense of this definition. Attribute values associated with an object cannot be modified and therefore cannot be a source of *dynamic modification* of a picture. Conversely, control values may be modified at any time and therefore are a source of dynamic modification

3.2.43**deferral mode**

a *state list* entry for the device which specifies the acceptable delay between the invocation of a CGI function and the final realization of its effect. Possible values are ASTI (At Some Time), BNI (Before Next Interaction) and ASAP (As Soon As Possible)

3.2.44**description table**

one of a set of tables giving “permanent” as opposed to “dynamic” characteristics of the *Virtual Device*. Contrast with *state list*. Description tables may in fact change by means outside the CGI, such as device reconfiguration by the operating system or, in some environments, by explicit *operator* or resource manager action (e.g. windowing systems). All information in the description table, including that indicating whether other information is variable, is inquirable

3.2.45**device class**

CGI *Virtual Devices* are classified, based on input and output characteristics, as INPUT, OUTPUT, or OUTIN. Devices of class OUTPUT have one active *drawing surface* which may or may not be the same as the active *display surface* depending on the device. Devices offering the functional capability defined in ISO/IEC 9636-6 may have several drawing surfaces. Devices of class INPUT have one or more *logical input devices* (LIDs), each with its own *measure*, but have no drawing or display surface. Devices of class OUTIN combine the drawing and display surfaces of the OUTPUT class and the LID facilities of the INPUT class

3.2.46**device coordinates (DC)**

a position specified by means of a device dependent coordinate system which is related to, but not necessarily identical with, the physical coordinate system by which a physical position on a display or input surface, such as a screen or tablet, is specified. Device coordinates may be in any metric — not necessarily uniform and with arbitrary origin. There may well be a final mapping outside the CGI from the “logical” device coordinates used within the CGI to the actual physical coordinates of a device (e.g. in a windowing system or an implementation supporting several CGI instances, i.e. different logical output devices, on the same physical resources)

3.2.47**device driver (implementation of the CGI)**

a procedurally bound (*see binding*) implementation which emits no CGI data stream encoding and which operates the physical device(s) by whatever means are necessary. This includes such examples as sending and receiving a non-CGI data stream, manipulating a frame buffer in host memory, and accessing control registers in an I/O bus peripheral

3.2.48**display surface**

the physical surface on which graphic information is presented to the *operator*. See *drawing surface*

3.2.49**downstream**

in the direction from *client* towards (virtual or real) device or *operator*

3.2.50**drawing mode**

this determines how colour or grey scale *pixel* values are combined during *rendering* of *graphic objects* and while performing *raster operation functions*. Drawing mode is either an *attribute value* of a particular graphic object, in which case it determines how the rendered information from that particular object is mixed into the *drawing surface*, or it is a parameter of a raster operation function, in which case it determines how the information from the one or two source *bitmap regions* is mixed into the destination bitmap region of the drawing surface. Drawing modes are composite values, the first component catering for logical combinations (BOOLEANOP), arithmetic combinations (ADDITIVEOP), and comparative combinations (COMPARITIVEOP) classes and the second component specifying a particular combination within the class

3.2.51**drawing surface**

the conceptual surface of the device on which the graphic picture is rendered or (for devices implementing the functional capability of ISO/IEC 9636-6) the currently selected *drawing bitmap* onto which both *graphic objects* are rendered and *raster operation functions* have their destination *bitmap region* specified. See *display surface*

3.2.52**dynamic modification**

change of a *control value* or a *segment attribute value* which may result in the rendered picture on the *drawing surface* being in some respects no longer a valid *rendering* of the content of *segment storage*. Note that *attribute values associated* with the *objects* in segments cannot be changed; so this is not a possible source of dynamic modification. Devices have different capabilities with respect to handling such dynamic modification. This capability is recorded for each possible cause of dynamic modification in the *description tables* under the entries dynamic modification accepted for <type of change>. Each such entry can have the values IMM (IMMediate), CBS (Can Be Simulated), or IRG (Implicit ReGeneration)

3.2.53**echoing**

echoing is the term used to describe feedback to the *operator* of the *measure* of a *logical input device*. Echoing may be provided by means of various styles referred to as echo types. The echo for a particular LID appears in a rectangular subspace referred to as the echo area on the *display surface* (as opposed to the *drawing surface*) of an OUTPUT or OUTIN device. Echoing is either automatically performed by the OUTIN device when requested by the *client* or, in the case of remote echoing on a device other than the one providing the *measure*, should be explicitly performed by the client using echo output facilities

3.2.54**echo request input**

a method of input by which the *client* can obtain the current *measure* of a *logical input device* in order to perform *remote echoing* on a different *Virtual Device*. This method returns data when the measure changes or on the occurrence of a trigger *firing*

3.2.55**edge**

the edge of a *fill object* consists of portions, each portion being initially the mathematical locus of a line *primitive* or the *boundary* of a fill primitive. The edge does not include any *implicit boundary portions* added during *closed figure* construction to maintain closure. Each portion of the edge has *associated* with it *edge attribute values* which can be different for different portions. After locus clipping, entire portions or pieces of portions lying outside the associated *clip rectangle* are discarded. *Clip boundary portions* are added to maintain closure; but these, like implicit boundary portions, are not edge portions. When the (locus clipped) edge is rendered, the remaining pieces of a portion (which may now be disjoint) are rendered using the edge attribute values associated with the portion. These may then be shape clipped to produce the final rendering of the edge. See also *boundary*, *interior*, *realized edge*, and *realized interior*

3.2.56**(data stream) encoding**

a (concrete) representation of an abstract functional specification in the form of a sequence of data items (e.g. characters, 8-bit bytes, etc.). The identification of each function and its parameter values are encoded to produce a *function representation*. Similarly, responses from an interpreter of the data stream are also encoded. Contrast with *binding*

3.2.57**error class**

errors which may occur when using CGI are classified into one of nine different error classes. For some of these, an error report will be entered into the *error queue*. The functional description specifies error reactions for the errors it specifically defines; but, non-standard errors are also provided for in the classification scheme. There are specific mechanisms for switching error detection and error reporting on or off on a per class basis

3.2.58**error queue**

a first-in-first-out (FIFO) queue, each entry of which consists of an error identifier and an error report. This queue can *overflow*. It cannot be *blocked* per se since a control equivalent to “blocking” is provided on a per class basis

3.2.59**escape function**

CGI provides an extension mechanism in the form of escape functions. There are three distinct categories: *generalized drawing primitive (GDP)*, ESCAPE and GET ESCAPE. GDPs provide for non-standard *primitives* and, like the standard *primitive functions*, are never *soliciting functions*. For those escapes which do not generate primitives, those which are soliciting functions are GET ESCAPES and the remainder are simply ESCAPES. Escape functions in all three categories may be registered in the International Registry of Graphical Items or are private to an implementation

3.2.60**event input**

a method of input in which an *input operation* (event) occurs asynchronously as a result of some *operator* action (see *firing* and *break action*) or *timeout* and for which an entry referred to as an **event report** is made in the *event queue*. *Logical input devices* have to be enabled for event input before they can contribute events to this queue

3.2.61**event queue**

there is a single event queue for all *logical input devices*. It is a first-in-first-out (FIFO) queue which has to be explicitly read by the *client*. The client does not receive any notification that an entry has been made. The queue can be *blocked* and it can also *overflow*. Once overflow has occurred, it is implicitly blocked until the client takes remedial action. Entries are made in the queue in time sequence of events which occur on those LIDs enabled for *event input*. Each entry is an *event report* which includes the LID class and index, the trigger index of the trigger which caused the event, a timestamp, the measure value, and the measure validity status

3.2.62**explicit boundary portion**

see *boundary*

3.2.63**external functions**

an external function is one which is not concerned with either the content or the control of the graphic information in the *Virtual Device*. MESSAGE is one such function defined in ISO/IEC 9636; but, some *escape functions* may also fall into the external category

3.2.64**fill object**

a general term for all (graphic) *objects* which have a (closed) boundary, an *edge*, and an interior. Fill objects are created from fill *primitives* and the *association* of edge attribute and interior (fill) *attribute values* or they are constructed as *closed figures* from line and fill primitives and *implicit boundary portions* generated to ensure closure. The *boundary* of a fill object is always closed even after *clipping* and the boundary encloses the *interior* (or clipped interior). The boundary may consist of several closed *region* boundaries; regions may be disjoint, intersect with, or be wholly contained within other regions. The definition of the interior of a fill object is determined by the *parity fill algorithm*. In general, the edge is rendered on top of the rendered interior; the boundary is not rendered. See also *realized edge* and *realized interior*

3.2.65**firing (a trigger)**

an action by the *operator* which indicates to a *logical input device* that something significant has occurred. After such an action, the *trigger* is said to have fired. A trigger cannot be fired unless it has been *armed*

3.2.66

font

a text primitive *attribute value* which is orthogonal to *character set* and determines the style of *rendering* of the individual glyphs

3.2.67

foreground colour (of objects)

the colour, determined by an *associated attribute value*, in which primitive components of *objects* or object *interiors* are rendered onto the *drawing surface*. If a line type is not continuous or a fill style is not solid, then *auxiliary colour* is used for the “gaps”. For text, foreground colour is used for the glyphs and auxiliary colour for the rest of the character cell

3.2.68

foreground colour (of mapped bitmaps)

a *control value* which determines the full depth value to which the foreground *pixels* of a *mapped bitmap* are expanded when they participate in *raster operation functions*

3.2.69

foundation profile

one of the basic *Profiles* defined in ISO/IEC 9636, (<input class> INPUT, 2-WAY OUTPUT, 1-WAY OUTPUT). All other Profiles should be defined to contain at least one of these Foundation Profiles

3.2.70

(CGI) function

the unit of communication between the *client* and the *Virtual Device*. This functional specification defines functions abstractly. *Bindings* and *encodings* will specify, for each function, a function representation — a concrete representation of the function and its parameters or of the request and the response. Functions are classified in various ways — see *attribute function*, *control function*, *external function*, *graphic function*, *inquiry function*, *primitive function*, *raster operation function*, and *soliciting function*. All functions request actions of various kinds from the Virtual Device. Soliciting functions require a response from the device

3.2.71

functionality

a contraction of “functional capability”; used to refer broadly to types of behaviour possible in a given context

3.2.72

generator

see *CGI generator*

3.2.73

generalized drawing primitive (GDP)

an *escape function* which generates a *primitive* or contributes to the construction of a *graphic object*

3.2.74

global attribute value

an *attribute value* which is *associated* with a (compound graphic) *object* as a whole, rather than with the individual *primitives* from which the object is constructed, is termed a global attribute value. Contrast with *local attribute value*

3.2.75

graphic function

this is a general term which covers all functions whose action is concerned with the content or control of the graphic information in the *Virtual Device*. Compare *external function* or *inquiry function*. Graphic functions include *attribute functions*, *control functions*, *primitive functions*, and *raster operation functions*

3.2.76**graphic object**

primitives, when they have had all their applicable *attribute values* associated with them, are referred to as (graphic) objects. There are special construction facilities which allow a (*compound*) *object* to be formed from several primitives with *local attribute values* and *global attribute values* associated with individual contributing primitives or with the object as a whole respectively

3.2.77**graphic object pipeline**

an abstraction of the collection of conceptual processes through which graphic (i.e. primitives and objects) information passes en route from the *client* to the *display surface* where it can be observed by the *operator*. *Raster operation functions* bypass most of this pipeline. Contrast with graphic pipeline which is used as an abstraction for all CGI information flowing (both ways) between the client and the operator

3.2.78**hatch style**

a particular representation for the *interior* of a *fill object* using fill style HATCH. A hatch style consists of one or more sets of (semi) regularly spaced lines in different directions. The exact appearance of hatch lines is implementation dependent

3.2.79**implementation-dependence**

where ISO/IEC 9636 recognizes that implementation variability is necessary, but cannot define a set of allowed behaviours, then the potential of variability is termed implementation-dependence. Contrast with *allowed latitude*

3.2.80**implicit boundary portion**

see *boundary*

3.2.81**implicit closure**

the process of ensuring that the *boundary* of a *fill object* remains closed even if the *client* does not explicitly create a closed boundary. Implicit closure is achieved by adding *implicit boundary portions* as necessary during the construction process

3.2.82**implicit segment regeneration mode**

a mode which determines whether automatic regeneration of the picture may occur or not (ALLOWED, SUPPRESSED, or UQUM) when some change of a *control value*, *segment attribute value*, or the contents of *segment storage* invalidates the current picture as an accurate *rendering* of the *objects* in segment storage. See *dynamic modification* and *quick update method*

3.2.83**indexed colour**

see *colour selection mode*

3.2.84**input class**

logical input devices are classified according to the type of data value (*measure*) they provide for return to the *client*. The classes defined are: LOCATOR, STROKE, VALUATOR, CHOICE, PICK, STRING, RASTER, or GENERAL. These are defined in detail in ISO/IEC 9636-5

3.2.85**input operation**

a general term for any sequence of *operator* and/or input device measure setting actions which makes an input value (*measure*) available for return to the *client*. The exact sequence of actions and the method of return of the measure to the client are determined by the input method. See *request input*, *sample input*, and *event input*

3.2.86

input surface coordinate (ISC) space

for input classes LOCATOR, STROKE, and PICK, there is a Cartesian coordinate system from which input points are specified by the *operator*. The space is referred to as the input surface coordinate (ISC) space. It is analogous to the *device coordinate space* for a *drawing surface*. A bounded rectangle within ISC space is termed the input surface. This is a subset of ISC space for which the physical device is capable of producing point locations. The *client* may define the mapping from ISC space to VDC space; this is termed the ISC-to-VDC Mapping. Each LID (for which it is relevant) may have its own ISC-to-VDC Mapping

3.2.87

inquiry function

a function which provides the *client* with access to information held in *description tables* or *state lists* of the *Virtual Device*. All inquiry functions are *soliciting functions*

3.2.88

interior (of a fill object)

the interior of a *fill object* is that part of *VDC space* “within” the *boundary* — “within” is defined by the *parity fill algorithm*. The interior may consist of several distinct or abutting subspaces. After *clipping*, the clipped interior is the set (which may now be a different set) of subspaces within the clipped boundary. See also *realized interior*

3.2.89

interpreter

see *CGI interpreter*

3.2.90

local attribute value

an *attribute value* which is *associated* with each individual *primitive* used in the construction of a (compound graphic) *object* rather than with the complete object is termed a local attribute value (Contrast with *global attribute value*)

3.2.91

logical input device (LID)

a logical component of an OUTIN or an INPUT device. LIDs are classified into one of the various *input classes*. A *Virtual Device* may have several LIDs which consist of a *measure* (input value), state information, and possibly associated *triggers*

3.2.92

mapped bitmap

see *bitmap*

3.2.93

measure

the term used for the abstract component of a *logical input device* representing the input value for that device which will be returned to the *client* on receipt of the appropriate *soliciting function* either directly from the LID or, in the case of *event input*, from the *event queue* where it will have been placed by the LID

3.2.94

message

a communication direct from the *client* to the *operator* which leaves no trace of its passage through any pipeline and is always displayed to the operator independent of whether the current display reflects the current *drawing surface*. Messages are character strings which are not subject to the *character set* or character coding announcer control which apply to text

3.2.95**modal value (modally set value)**

a value is referred to as modal if it is set by a specific function and recorded in a *state list*. That value is then applicable, whenever relevant, to subsequent functions until such time as it is reset by another use of the specific setting function or, in certain cases, it is reset to default by the *Virtual Device* as a result of VDC TYPE changes. Contrast with parameter value, which is always local to a particular function invocation

3.2.96**non-dissociable trigger**

a *trigger* whose association with a *logical input device* is an inherent characteristic of the device and cannot be overridden by the *client*. See *trigger association*

3.2.97**non-retained data**

any rendered object information resulting from *objects* passed down the *graphic object pipeline* which were not stored in *segments* (i.e. while no segment was open) as well as all information resulting from *raster operation functions* which is on the *drawing surface* is referred to as non-retained data. In other words, any information on the drawing surface which cannot be reproduced by the sequence — PREPARE DRAWING SURFACE, set all segment visibility attributes to VISIBLE, REDRAW ALL SEGMENTS — is non-retained data

3.2.98**object**

see *graphic object*

3.2.99**object clipping mode**

a *control value* which determines how the process of *clipping* (graphic) *objects* is conducted. This is either on the mathematical locus of the object before the application of *rendering attribute values* (LOCUS), on a rendered shape after their application (SHAPE), or on first the one and subsequently the other (LOCUS THEN SHAPE). The three settings can all give different effects. There are separate clipping mode control values for each of lines, markers, and edges

3.2.100**one-way (1-way) output**

this describes the situation where the *client* never uses any *soliciting functions*. All communication across the CGI is from client to *Virtual Device* and there is never any response from device to client

3.2.101**operator**

the human user of the CGI *Virtual Device* realization. Contrast with *client*. The operator may interact with the client by observing the realized output and by providing input (related or unrelated to the output), responding to *messages*, etc.

3.2.102**overflow (of a queue)**

some implementations may limit the space of a queue to some pre-set number of entries (fixed or indeterminate depending on whether the size of the entry is fixed). Other implementations may use dynamic allocation from a resource pool used for other purposes as well as the queue entries. When queue space is exhausted, the next attempted entry causes the queue to overflow. Overflow of the *event queue* results in the latest entry always being overwritten with an “error reports lost” error which includes an updated count of entries lost. Overflow of the *input queue* results in the queue becoming implicitly *blocked* and it remains so until the *client* takes remedial action

3.2.103**parity fill algorithm**

this algorithm determines the *interior* of a *fill object* as that set of points not lying on the *boundary* for which all lines to infinity cross the boundary an odd number of times

3.2.104**pattern (of fill pattern)**

one of the styles for *interior* fill of a *fill object* is pattern fill. A pattern is a two-dimensional uniform array of *cells* each specifiable as having a different colour or grey scale value. (Compare with CELL ARRAY which is similar.) Patterns are selected by the *client* by means of the pattern index (object) *attribute value*

3.2.105**pattern (of bitmap region)**

a *bitmap region* used as one of the source parameters in a 3-operand *bitblt* or specified by the FILL BITMAP function for rendering with interior style BITMAP is referred to as a pattern. In either case, the pattern (bitmap region) is replicated as a tile. For the 3-operand bitblt case, the pattern is replicated to build a region matching the size of the other source region. In the bitmap fill case, the pattern is replicated as necessary to fill the *interior*

3.2.106**picking**

conceptually, this is an operation which allows the *operator* to point at something on the *display surface* after which the *Virtual Device* responds to the *client* with an indication of what the operator pointed to. This feature works in conjunction with segment capability. The response lists data structures which indicate certain *segments* which would be drawn sufficiently close to where the operator pointed. The precise definition of the picking behaviour is in terms of the contents of *segment storage*, and it is the responsibility of the client to make sure that the visible display is relevant for the operator

3.2.107**pixel**

in the context of raster devices, the smallest element of the *drawing surface* of a *Virtual Device* to which an individual colour or grey scale value can be assigned. The assigned value is a device dependent pixel value and *rendering of graphic objects* and all *raster operation functions* act directly on pixel values so combination characteristics may vary from device to device. Pixels may lie on or between addressable device points as specified by an entry in a *description table*

3.2.108**primitive**

a primitive gives the geometric specification for some portion of a picture. Primitives are passed across the CGI interface by the *client*, using *primitive functions*. Primitives do not carry any *attribute values*. These are *associated* with them later in the *graphic object pipeline* to form *objects*

3.2.109**primitive function**

a primitive function is one whose primary action is to generate a *primitive*. Primitive functions are classified according to *primitive type*

3.2.110**primitive type**

one of “line”, “marker”, “text”, “fill”, or “image”. A primitive type is an adjective which can apply to “function”, “primitive”, “object”, “aspect”, or “attribute”. (E.g. “fill function”.) Note the anomalous word “edge” which can be used as a type for “aspect” or “attribute”, even though “edge” is not itself a primitive type. *Generalized drawing primitives* may generate *primitives* of various types and be subject to *aspects* associated with those types

3.2.111**profile**

a specification of a particular collection of CGI functions and resource requirements as well as any relevant constraints or *escapes* required to meet the needs of a particular set of users, sometimes referred to as a constituency. There are Foundation Profiles and Constituency Profiles. Some Constituency Profiles are defined in ISO/IEC 9636; others may be registered

3.2.112**prompting**

an action, enabled or disabled by the *client*, then taken internally by a *logical input device* at the beginning of an *input operation* to draw the *operator's* attention to the fact that an input action is expected. In a *remote echoing* situation, prompting has to be specifically performed by the client. Prompting may take various styles known as prompt types selectable by the client

3.2.113**quick update method**

in the context of *implicit segment regeneration*, implementation-dependent methods of modifying the picture, in response to certain changes, are permitted for the sake of speeding up the display process. Such methods need not be completely faithful to the segment model. For example, *segments* may be “erased” by “drawing” them in background colour; segment display may be done without regard to display priorities. Such methods may be employed by an implementation only when explicitly allowed by the *client* when *implicit segment regeneration mode* is UQUM (Use Quick Update Method), and only for changes whose *dynamic modification accepted for entry* is CBS (Can Be Simulated)

3.2.114**raster operation functions**

functions which request operations directly on the *pixel values* in *bitmaps* on devices offering the functional capability defined in ISO/IEC 9636-6 are referred to as raster operation functions. They consist of the PIXEL ARRAY functions and the *bitblt* functions. The specific action of these functions is controlled by parameters rather than *state list* values for relevant factors such as *transparency* and *drawing mode*

3.2.115**realized edge**

this is what is rendered after the associated *attribute values* (width, type, colour, etc.) have been applied to the *explicit boundary portion* of a *fill object*. The exact impact of clipping is controlled by the *edge object clipping mode*. The realized edge is rendered “in front of” the *realized interior* and may therefore obscure or modify part of that interior. See *boundary*

3.2.116**realized interior**

this is what is rendered by applying associated fill *attribute values* and any applicable object clipping to the *interior* of a *fill object*. For all fill styles except HOLLOW, the interior is realized first. The *edge* is then realized and overlaid or mixed depending on *auxiliary colour*, *transparency*, and *drawing mode* (or on device overwrite capability). For interior style HOLLOW, the *boundary*, including the *clip boundary portion*, is rendered using the associated fill colour and implementation-dependent “boundary attributes” and the realized edge is then overlaid. This is the only case when the boundary is rendered; for all other interior styles, implicit boundary portions and clip boundary portions are not rendered

3.2.117**regeneration**

see *implicit segment regeneration mode*

3.2.118**region (of a fill object)**

each *fill object* consists of one or more regions. Each region has its own closed *boundary* which may have been formed using *implicit closure* and modified by object *clipping*. Regions may have self-intersecting boundaries and may overlap or abut other regions. Regions are formed explicitly by use of the NEW REGION function or implicitly when a *fill primitive* is used during *closed figure* construction. Each closed polygon of a POLYGON SET is a region. The only fill objects which may have multiple regions are closed figures, POLYGON SETs and possibly GDPs

3.2.119**region (of a bitmap)**

a rectangular subset of a *bitmap* used as a source or a destination in *raster operation functions* or as a *pattern* for interior style BITMAP

3.2.120**remote echoing**

in situations in which a *Virtual Device* realization does not have the capability to do *echoing* (locally) within the device (e.g. when the input device and the output device are supported by different CGI instances), the *client* may simulate the echoing process by making use of some input method (possibly *echo request input*) on one *logical input device* and echo output on a different device. Such echoing performed explicitly by client action is referred to as remote echoing. Echo output facilities also cater for remote *acknowledgement* and remote *prompting*

3.2.121**rendering**

the collection of processes (*downstream of segment storage* in the *graphic object pipeline*) which convert the geometric specification of an *object* and its associated *attribute values* to the form in which it will be presented on the *drawing surface*. The term generally covers transformations, application of attribute values, and *clipping*, as well as processes specifically shown as locus, shape, and physical rendering. All processing concerned with rendering may involve varying degrees of approximation which may be device-dependent and generally will not be inquirable

3.2.122**request input**

a method of input in which the *measure* value of a *logical input device* is returned synchronously to the *client* when the *operator* fires a trigger (see *firing*). All functions concerned with request input are *soliciting functions* and, having interpreted one, the *Virtual Device* will suspend all further interpretation until the operator has taken action or a *timeout* has occurred and a response is delivered to the client. Contrast with *event input*, *sample input*, and *echo request input*

3.2.123**sample input**

a method of input in which the current *measure* of a *logical input device* is returned immediately to the *client* when an appropriate input *soliciting function* is invoked. No *operator* action is required. Contrast with *request input* and *event input*

3.2.124**segment**

a grouping of (graphic) *objects* retained in the *Virtual Device* in a virtual resource referred to as segment storage. The *client* controls the existence and content of each segment. (See *segment definition*.) Each segment has its own unique, but changeable, segment identifier and carries an *associated* set of segment attribute values which determines its detectability, display priority, highlighting, pick priority, segment transformation, and visibility. Segment attribute values apply to the entire contents of the segment. These values may be changed by the client at any time and are thus a potential source of *dynamic modification*

3.2.125**segment definition**

the process of constructing the segment's contents (as opposed to setting its *segment attribute values*). Only the open segment can be undergoing definition. The process is initiated by CREATE SEGMENT and terminated by CLOSE SEGMENT though an existing segment can be subsequently reopened and its definition continued (sometimes referred to as appending to a segment). When a segment is open, *objects* flowing along the *graphic object pipeline* are entered into the open segment, thus defining its content

3.2.126**skewed**

non-perpendicular. The term is used in several contexts — to refer to text when the orientation vectors are non-perpendicular, to refer to *cells* of CELL ARRAY or *patterns* when the defining points form a non-rectangular parallelogram, and so on. Transformations may also produce skewing