

INTERNATIONAL STANDARD

ISO/IEC
9636-4

First edition
1991-12-15

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

Part 4: Segments

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

Partie 4: Segments



Reference number
ISO/IEC 9636-4:1991

AVA Documents
Document Sharing Hub

Contents

	Page
Foreword.....	iv
Introduction.....	v
1 Scope	1
2 Normative references	2
3 Concepts.....	3
3.1 Introduction.....	3
3.1.1 Relationship of CGI segments to the graphic output pipeline	3
3.2 Creating segments.....	3
3.2.1 Segment identifiers	3
3.2.2 Creating and closing segments.....	3
3.2.3 Non-retained data.....	4
3.2.4 Segment storage overflow.....	4
3.3 Segment attributes	4
3.3.1 Introduction	4
3.3.2 Segment highlighting.....	5
3.3.3 Segment visibility.....	5
3.3.4 Segment detectability	5
3.3.5 Segment display priority	5
3.3.6 Segment pick priority.....	5
3.3.7 Segment transformation	6
3.4 Segment display.....	6
3.4.1 Introduction	6
3.4.2 Segment regeneration	6
3.4.3 Quick update methods.....	8
3.4.4 Explicit segment display	9
3.5 Copy segment and the inheritance filter.....	9
3.6 Delete and rename segments	10
3.7 Inquiry	10
3.8 Picking	10
3.9 State restrictions.....	11
4 Interactions with other parts of ISO/IEC 9636	12
4.1 Interactions with ISO/IEC 9636-2.....	12
4.1.1 INITIALIZE and TERMINATE	12
4.2 Interactions with ISO/IEC 9636-5.....	12
4.3 Interactions with ISO/IEC 9636-6.....	12
5 Abstract specification of functions.....	13
5.1 Introduction.....	13
5.1.1 Data types employed.....	13
5.1.2 Validity of returned information.....	13
5.2 Segment manipulation functions	13
5.2.1 GET NEW SEGMENT IDENTIFIER	13
5.2.2 CREATE SEGMENT	13
5.2.3 REOPEN SEGMENT	14
5.2.4 CLOSE SEGMENT	14
5.2.5 COPY SEGMENT	15
5.2.6 DELETE SEGMENT.....	16
5.2.7 DELETE ALL SEGMENTS.....	16
5.2.8 RENAME SEGMENT.....	17
5.2.9 DRAW ALL SEGMENTS.....	17

© 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

	5.2.10	IMPLICIT SEGMENT REGENERATION MODE	17
	5.2.11	RESET REGENERATION PENDING	18
	5.2.12	PICK IDENTIFIER	18
5.3		Segment attribute functions	18
	5.3.1	SEGMENT VISIBILITY	18
	5.3.2	SEGMENT TRANSFORMATION	19
	5.3.3	SEGMENT HIGHLIGHTING	19
	5.3.4	SEGMENT DISPLAY PRIORITY	19
	5.3.5	SEGMENT DETECTABILITY	20
	5.3.6	SEGMENT PICK PRIORITY	20
5.4		Miscellaneous segment functions	20
	5.4.1	SIMULATE PICK	20
	5.4.2	INHERITANCE FILTER	21
	5.4.3	CLIPPING INHERITANCE	23
6		Segment inquiry functions	24
	6.1	Introduction	24
	6.1.1	Data types employed	24
	6.1.2	Validity of returned information	24
	6.2	Segment description table	24
	6.2.1	INQUIRE SEGMENT CAPABILITY	24
	6.3	Segment state list	25
	6.3.1	INQUIRE SEGMENT STATE	25
	6.3.2	INQUIRE LIST OF INHERITANCE FILTER SETTINGS	25
	6.3.3	INQUIRE CLIPPING INHERITANCE	25
	6.3.4	INQUIRE LIST OF SEGMENT IDENTIFIERS IN USE	25
	6.4	Individual segment state list	26
	6.4.1	INQUIRE INDIVIDUAL SEGMENT STATE	26
7		CGI description tables and state lists	27
	7.1	Description tables	27
	7.2.1	Segment state list	28
	7.2.2	Individual segment state list	28
A		Formal grammar of the functional specification	29
B		Segment errors	39
C		Guidelines to implementors	40
C		Examples of COPY SEGMENT	41

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 9636-4 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 and D are for information only.

Introduction

This part of ISO/IEC 9636 describes the functions of the Computer Graphics Interface concerned with segment storage.

The functionality incorporated in this part of ISO/IEC 9636 is concerned with creating, modifying, and manipulating graphic pictures using segments.

The functionality described in this part of ISO/IEC 9636 pertains to Virtual Devices of class OUTPUT and OUTIN.

This page intentionally left blank

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

Part 4: Segments

1 Scope

This part of ISO/IEC 9636 defines those functions of the Computer Graphics Interface concerned with the creation, modification, and manipulation of graphic pictures using segments.

This part of ISO/IEC 9636 is part 4 of ISO/IEC 9636, and should be read in conjunction with ISO/IEC 9636-1, ISO/IEC 9636-2, and ISO/IEC 9636-3. 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 Virtual Devices of class OUTPUT and OUTIN.

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 7942 : 1985 *Information processing systems – Computer graphics – Graphical Kernel System (GKS) 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 9636-3 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 3: Output.*

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.*

¹⁾ To be published.

3 Concepts

3.1 Introduction

This part of ISO/IEC 9636 specifies how graphic objects may be grouped in segments and be identified by a unique segment identifier. The graphic objects stored in segments are composed of graphic primitives with associated attribute values. This part of ISO/IEC 9636 defines a set of functions for creating, modifying, and manipulating segments. This functionality is divided into the following areas:

- *Segment manipulation functions*, including segment creation, deletion, renaming, and copying. Graphic objects may also be appended to the end of an existing segment. Graphic objects within a segment, however, cannot be modified or deleted.
- *Segment attribute functions*, including modification of segment attributes (e.g. transformation, visibility). Some segment attributes affect the rendered appearance of segments and can be a basis for feedback during graphic picture manipulation. Other attributes affect how input concepts and segments are associated in support of pick input.
- *Segment inquiry functions*, by which access is provided to the information in the description tables and state lists concerned with segments.

3.1.1 Relationship of CGI segments to the graphic output pipeline

The overview of the CGI Graphic Object Pipeline presented in ISO/IEC 9636-1, clause 5, describes the relationship between segments and the pipeline. Additional details are described in this part of ISO/IEC 9636 in connection with the particular functional areas that are affected.

3.2 Creating segments

3.2.1 Segment identifiers

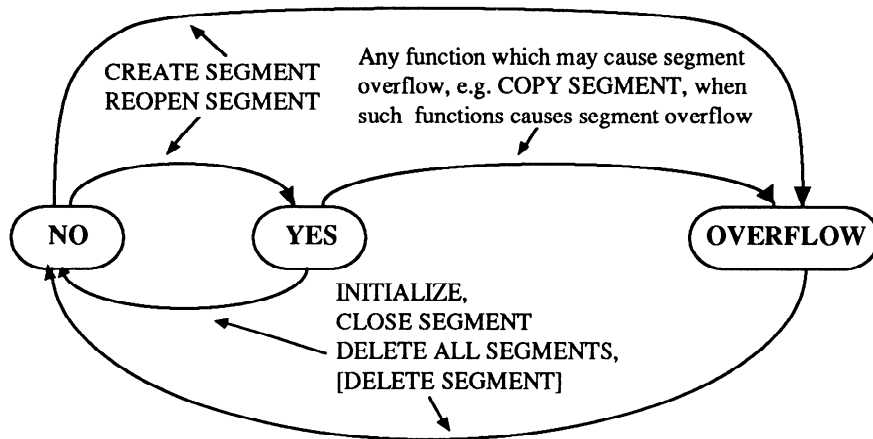
Each segment has a unique segment identifier associated with it. The client can refer to a specific segment by using its associated segment identifier. Once a segment is deleted, the identifier that was associated with it may be reused for another segment definition. The segment identifier associated with a segment may be changed with the RENAME SEGMENT function. The Maximum Number of Simultaneously Existing Segments entry in the Segment Description Table indicates the number of segments which may coexist at one time (this number is implementation-dependent). The List of Segment Identifiers in use may be inquired from the Segment State List.

The client provides a segment identifier as a parameter to the CREATE SEGMENT function used to create a new segment. For cases in which the client does not wish to keep a record of which identifiers are available, the function GET NEW SEGMENT IDENTIFIER is provided. This function will return an available segment identifier. A segment identifier is a parameter of most of the functions defined in this part of ISO/IEC 9636.

3.2.2 Creating and closing segments

Segments are created when the CREATE SEGMENT function is invoked. This is the only means of creating a segment. A segment is opened by CREATE SEGMENT and the Segment Open State entry in the Segment State List is set to YES. There can be at most one segment open at any one time. While a segment is open, graphic objects passed along the Graphic Object Pipeline are stored in the segment (a process termed segment definition) and rendered on the drawing surface if the segment's Visibility attribute is VISIBLE. The open segment is closed by invoking the function CLOSE SEGMENT and the Segment Open State entry in the Segment State List is set to NO (see figure 1).

An existing segment may be reopened at a later time with the REOPEN SEGMENT function. When reopened, subsequent graphic objects are appended to the open segment using the same conceptual mechanism as when the segment was created. A reopened segment is closed by invoking the function CLOSE SEGMENT.



NOTE – DELETE SEGMENT will change the state from YES or OVERFLOW to NO only if the currently open segment is deleted.

Figure 1 – Segment Open state transitions

Note that objects stored in segments have associated with them all applicable individual attribute values independent of the value of the corresponding ASF value. Thus by changing the ASF (by means of INHERITANCE FILTER, see 3.5), it is still possible to display the object with the individual attribute value associated on creation even when the corresponding ASF value was initially BUNDLED.

3.2.3 Non-retained data

Any graphic object passing along the pipeline when there is no segment open is termed non-retained data. Such graphic objects are rendered onto the drawing surface in the usual manner as described in ISO/IEC 9636-3. However, non-retained data is never rendered again as a result of implicit or explicit regeneration.

3.2.4 Segment storage overflow

Segment storage overflow may occur when storing graphic objects in segments. If overflow occurs, all graphic objects prior to the first object that causes the overflow are stored in the segment. The graphic object causing the overflow as well as graphic objects following the segment storage overflow and prior to segment closure are rendered as non-retained data and a class 6 error is generated.

Once an overflow has occurred, the Segment Open State in the Segment State List is set to OVERFLOW. Invoking CLOSE SEGMENT, DELETE SEGMENT, or DELETE ALL SEGMENTS is the only way to change the Segment Open State from OVERFLOW to NO (other than by INITIALIZE) (see figure 1).

Segment storage overflow may also occur when a segment is created or reopened. In this case, the CREATE SEGMENT or REOPEN SEGMENT function is ignored, a class 6 error is generated, and graphic objects following the segment storage overflow and before the segment is closed are rendered as non-retained data.

3.3 Segment attributes

3.3.1 Introduction

The segment attribute values associated with a particular segment affect that segment's rendering and pick input properties. A segment's attribute values, including Segment Transformation, are set by default when the segment is initially created and can be changed individually at any time while the segment still exists. Segment attribute values may be inquired from the Individual Segment State Lists.

The rendering of graphic objects within a segment while the segment is being defined depends on the segment's attributes of Visibility, Highlighting, Display Priority, and Segment Transformation, and the current setting of the Implicit Segment Regeneration Mode. For example, if Visibility is set to INVISIBLE before the creation of the first graphic object to be stored in the segment, the segment is not rendered. If Visibility is changed part way through the segment definition and the Implicit Segment Regeneration Mode is SUPPRESSED, each new object will be rendered as it is received without regard to the Display Priority attribute.

3.3.2 Segment highlighting

The Highlighting attribute of a segment can take the values NORMAL or HIGHLIGHTED. The form in which highlighting is represented is implementation-dependent. However, the appearance of a segment when highlighted shall be different from when it is not. When a segment's highlighting attribute is changed, all of the graphic objects of the segment are displayed based on the Implicit Segment Regeneration Mode and the segment's Display Priority. (See 3.4.)

3.3.3 Segment visibility

The Visibility attribute of a segment can take the value of INVISIBLE or VISIBLE. When a segment's Visibility attribute is set to be VISIBLE, all of the graphic objects of the segment are displayed based upon the Implicit Segment Regeneration Mode and the segment's Display Priority. A segment can be picked if and only if it is both VISIBLE and DETECTABLE. If segment storage overflow occurs, objects following the segment storage overflow and before the segment is closed will be rendered as non-retained data independent of the value of the Visibility attribute.

3.3.4 Segment detectability

The Detectability attribute of a segment can take the value of UNDETECTABLE or DETECTABLE. Segment detectability does not affect the display or appearance of segments. A segment can be picked if and only if it is both VISIBLE and DETECTABLE.

3.3.5 Segment display priority

The Display Priority attribute of a segment determines how overlapping segments are displayed. In general, segments with higher display priorities will be displayed as if they were in front of segments with lower display priorities. This part of ISO/IEC 9636 allows latitude in cases where overlapping segments have the same display priority. The preferred behaviour is as follows:

The effective relative display priority of a set of overlapping segments having the same display priority value is consistent with their relative creation time order. In particular, the most recently created segments will appear in front of segments created earlier (this is consistent with the effect of drawing the same graphic objects as non-retained data in the same order as the segments were created). Reopening a segment does not change the effective relative display priority of the segment.

The Effective Relative Display Priority entry in the Segment Description Table indicates the supported capability, as follows:

- A level of TIME ORDER indicates that the most recently created segment is displayed in front, as described above;
- A level of TIME ORDER - REOPEN is similar to TIME ORDER, except that a change in the effective relative display priority is possible for a segment if subsequently reopened;
- A level of OTHER indicates that there is no guaranteed effective relative display priority behaviour.

The SEGMENT DISPLAY PRIORITY function is used to change the display priority of a segment. When a segment's Display Priority is changed, the segment's display is based on the Implicit Segment Regeneration Mode and the new Display Priority. (See 3.4.)

3.3.6 Segment pick priority

The Pick Priority attribute of a segment is used to resolve the picking of segments which overlap. If two or more segments overlap and the pick location is within the intersection of these segments, the segments picked will be one, or more, of those with the highest pick priority. The returned list of pick values (see 5.4.1 and PICK input in ISO/IEC 9636-5) corresponds to all those segments having the highest identical pick priority and are returned in order of decreasing display priority.

3.3.7 Segment transformation

The Segment Transformation attribute specifies a coordinate transformation to be applied to the graphic objects contained in the segment. This transformation allows scaling, translation, and rotation of segments. The segment transformation is a transformation of VDC space to VDC space and is distinct from the VDC-to-Device Mapping (see ISO/IEC 9636-2, 3.3), which is a transformation of VDC space to DC space.

The Segment Transformation attribute is set by the SEGMENT TRANSFORMATION function. This function simply replaces the Segment Transformation attribute associated with a particular segment. A segment transformation is composed of a 2x2 scaling and rotation portion and a 2x1 translation portion. The default Segment Transformation for each segment is the identity transformation.

The Segment Transformation attribute is associated with the graphic objects of a segment when they enter into the graphic object pipeline. This associated transformation is then applied during the rendering of the graphic objects.

A graphic object is transformed by the transformation resulting from the concatenation of the segment transformation and the object's transformation attribute before the application of the VDC-to-Device Mapping. The method of concatenation is such that the final effect during rendering is as if the object's transformation attribute was applied first, followed by application of the segment transformation. This transformation is applied to the graphic object's VDC point data and all associated attributes having VDC parameterizations, with the exception of the associated clip rectangle.

NOTE – If the client wishes to accumulate or concatenate transformations for the purpose of setting the Segment Transformation attribute, it is necessary to do this above the CGI. The use of segment transformations may produce coordinates that cannot be expressed within the VDC range; this is handled in an implementation-dependent way.

3.4 Segment display

3.4.1 Introduction

Segment display is the process which produces a visible image on a drawing surface from the graphic objects in a segment. The functions COPY SEGMENT and DRAW ALL SEGMENTS provide for the display of segments, individually and collectively. Segments may also be displayed implicitly, under some circumstances, without using the above functions (see 3.4.2).

In addition to the attributes associated with the graphic objects stored in segments, the Highlighting, Visibility, Display Priority, and Segment Transformation attributes of segments affect the appearance of the displayed segment.

Some devices cannot immediately change a picture. A plotter for example can only add to a picture; the plotter would need to advance the paper and redraw the picture to show the effect of a change of Segment Transformation. Similarly, a change of a segment's Display Priority attribute might cause different sections of a picture to become visible or obscured, and Visibility and Highlighting attributes may also cause the picture to alter.

On devices which cannot erase or change part of a picture, Implicit Segment Regeneration Mode of SUPPRESSED is more efficient in terms of time and material. Implicit Segment Regeneration Mode in combination with the Visibility attribute gives the client the ability to accumulate picture changes. At some later time, the client can explicitly utilize PREPARE DRAWING SURFACE and DRAW ALL SEGMENTS.

3.4.2 Segment regeneration

Conceptually, the contents of segment storage, along with the values of various state list and bundle table entries, describe a certain picture. In addition to the stored graphic objects, the state list information which may affect the appearance of the picture is shown in table 1.

Segment creation or deletion, as well as dynamic changes to any of the state list entries in table 1, may affect the accuracy of the rendered picture (i.e. whether or not the current, rendered picture representation accurately reflects the picture described in segment storage and state lists). The process of bringing the rendered picture representation to a state consistent with the current contents of segment storage and state lists is referred to as *regeneration*.

Table 1 – State list information which may affect the appearance of the picture

VDC-to-Device Mapping	Segment Attributes:
Colour table entries	Display Priority
Background Colour	Visibility
Bundled primitive attribute values (depending on associated ASFs)	Highlighting
Pattern table entries	Segment Transformation
List of Font Names	

When there is a discrepancy between the current contents of the drawing surface and the described picture, a *regeneration is pending*. The Segment State List includes a Regeneration Pending entry, which indicates whether or not any such discrepancy currently exists. Changes to the picture, or to the state list information that controls the appearance of the picture (see table 1) may cause the Regeneration Pending entry to be set to YES. If there is any visible picture discrepancy, the implementation shall set the entry to YES. The implementation may also set the entry to YES in case that a change effects no visible discrepancy, but the implementation is not able to determine that there is no discrepancy. Once set, Regeneration Pending remains set until an implicit regeneration or until the function RESET REGENERATION PENDING has been invoked.

There are a number of CGI functions which may change the contents of the drawing surface in such a way that it no longer accurately reflects the contents of segment storage, but do not cause Regeneration Pending to be set. This set of functions includes COPY SEGMENT when there is no segment open and all graphic primitive functions executed when there is no segment open (thereby creating non-retained data). It also includes bitblts, PIXEL ARRAY, DRAWING BITMAP, and PREPARE DRAWING SURFACE. It is the client's responsibility to be aware of the implications of the use of these functions in conjunction with segments and handle any desired regeneration explicitly.

For performance reasons, it may be preferable to batch a number of picture changes and then perform a single regeneration. A means to control regeneration behaviour is provided by the Implicit Segment Regeneration Mode entry in the Segment State List. When Implicit Segment Regeneration Mode is SUPPRESSED, the Virtual Device will not perform any regeneration even if one is pending.

For particular devices, some changes may not necessitate a regeneration. An important (and common) example concerns changes to the Colour Table on a raster device. Typically, such devices implement a colour table in hardware. Changing the colour representation associated with a given colour index immediately (and retroactively) affects all pixels drawn with that index. No regeneration process is required to bring the picture up to date.

For each state list entry corresponding to information in table 1, a Dynamic Modification Accepted For entry in a description table indicates which changes

- can be performed immediately (IMM);
- can be simulated (CBS) (see 3.4.3);
- lead to an implicit regeneration (IRG).

If the Dynamic Modification Accepted For <type of change> entry is IMM, the effect of a change to the relevant state list information will propagate immediately to the picture without any regeneration.

If the Dynamic Modification Accepted For <type of change> entry is IRG and Implicit Segment Regeneration Mode is SUPPRESSED, the effect of a change to the relevant state list information will be to set Regeneration Pending to YES but no regeneration will be initiated.

If the Dynamic Modification Accepted For <type of change> entry is IRG and Implicit Segment Regeneration Mode is ALLOWED, any change of the relevant state list information will initiate the regeneration process immediately after the function that caused the change of state list information.

If the Dynamic Modification Accepted For Adding Objects To Open Segment entry in the Segment Description Table is IRG and the Implicit Segment Regeneration Mode is SUPPRESSED, then each object added to the open segment will be rendered as it is received without regard to the Display Priority attribute value; this will cause the Regeneration Pending entry to be set to YES if there is any visible picture discrepancy. If the Implicit Segment Regeneration Mode is ALLOWED, then each object added to the open segment shall be rendered with the correct display priority (relative to possibly overlapping segments of higher display priority).

Other operations, beside changing the state list entries, may cause a discrepancy between the described picture and the drawing surface. These include deleting a segment and adding additional graphic objects to an open segment. The Segment Description Table has Dynamic Modification Accepted For flags for these changes. The distinction is also made between Visibility changes from visible to invisible and from invisible to visible.

If a regeneration is pending at the time of a transition of the Implicit Segment Regeneration Mode to ALLOWED, then a side effect of the IMPLICIT SEGMENT REGENERATION MODE function is to cause a regeneration to occur at that time.

For any function that causes a regeneration to occur, the regeneration process occurs after execution of the function and subsequent functions will not be interpreted until after the effects of the regeneration have been realized. Conceptually, the effect of a regeneration may be described as follows:

The drawing surface is prepared (cleared), and copies of all of the graphic objects in all the visible segments in segment storage are inserted into the Graphic Object Pipeline as described in ISO/IEC 9636, and the Regeneration Pending entry is set to NO.

All non-retained data is erased if a regeneration occurs. The prediction of regeneration requires careful attention to the Dynamic Modification Accepted For flags, which may differ from one CGI implementation to another.

Regeneration should not be confused with Deferral Mode which controls an entirely different aspect of Virtual Device behaviour. One of the more important manifestations of deferral relates to buffering that may occur between calling a CGI function at a procedural interface and subsequent interpretation of that function at a data stream interpreter further downstream in the implementation. Thus deferral is concerned with the promptness with which function invocations will be acted upon. Regeneration effects never occur until the causing functions are actually acted upon.

3.4.3 Quick update methods

When segments are complex or there are many of them, regeneration may be a fairly time consuming process. The CGI provides a way to allow implementations to offer quick update methods for changes that would otherwise require a regeneration. Examples might include changes to segment attributes, such as a change in Segment Transformation, Visibility, Highlighting or Display Priority. Quick Update Methods may be used to gain performance in interactive situations at the risk of the picture no longer accurately reflecting the state of segment storage.

For example, a segment may be “erased” by drawing it in the background colour. In cases where this operation is feasible, performance might be significantly improved, especially when dealing with a large number of segments. Many clients may accept a compromise in picture accuracy in favour of greater interactivity. At some appropriate later time (perhaps, upon operator request), the client can regenerate an uncompromised picture using DRAW ALL SEGMENTS.

The above type of picture repair is referred to as using a *quick update method*, as opposed to *regeneration*. Quick update methods are not claimed to be faithful to the segment model, but under many circumstances, are guaranteed to be faster. For many applications, there may be circumstances under which the client can know that the quick update will actually be perfect. For example, if the client never draws any overlapping segments.

Changes which have implementation defined quick update methods are indicated by a Dynamic Modification Accepted For <type of change> flag of value CBS. This indicates that the effect of such a change can be simulated. Any change which can be simulated can also be regenerated.

Quick update behaviour is permitted to occur if the Implicit Segment Regeneration Mode is set to UQUM, (for Use Quick Update Method) and the Dynamic Modification Accepted For <type of change> flag is CBS. If Implicit Segment Regeneration Mode is set to UQUM, regeneration remains suppressed.

The only point in time when a quick update can occur is when the function causing the need for it is invoked. If Implicit Segment Regeneration Mode is SUPPRESSED, then quick update methods are also suppressed. If the Implicit Segment Regeneration Mode is either SUPPRESSED or UQUM and a change is made for which Dynamic Modification Accepted For entry is CBS, then Regeneration Pending will be set to YES. An exception to this may occur when a particular implementation knows that the picture repair using quick update was, in fact, faithful to the segment model and state lists.

If the Implicit Segment Regeneration Mode is ALLOWED and a change is made for which the Dynamic Modification Accepted For flag is CBS or IRG, then an implicit segment regeneration will be performed as a result of the function causing the change.

Although control of the use of quick update methods is provided for in ISO/IEC 9636-4, the actual behaviour of an instance of a quick update is implementation-dependent.

3.4.4 Explicit segment display

The CGI functions COPY SEGMENT and DRAW ALL SEGMENTS may be used to explicitly display segments. DRAW ALL SEGMENTS displays all visible segments in a manner consistent with their display priorities. Its operation is independent of the Regeneration Pending state list entry and does not affect the setting of this entry. A single segment may be displayed by invoking the COPY SEGMENT function with the identity transformation as well as the INHERITANCE FILTER function with the filter selection list set to ALL and the selection setting set to SEGMENT. This segment will be rendered in front of any already rendered segments with no regard to display priority.

3.5 Copy segment and the inheritance filter

The COPY SEGMENT function copies the graphic objects of an identified source segment into the Graphic Object Pipeline, concatenating a coordinate transformation specified by the copy transformation parameter and optionally concatenating the source segment's Segment Transformation attribute with the transformation associated with the copied objects. The copy transformation affects all associated clip rectangles; the source segment's Segment Transformation does not affect clip rectangles.

The copied graphic objects may be altered in a variety of ways:

- a) The inheritance filter controls whether individual attribute values of the graphic objects are re-associated from the CGI state lists (see 5.4.2).
- b) The CLIPPING INHERITANCE function controls whether the effective clip region associated with the object is ignored or intersected with the Clip Rectangle in the General Attributes and Output Control State List.
- c) A copy transformation is concatenated with the graphic objects' associated transformation and, optionally, the Segment Transformation of the source segment may also be concatenated according to the rules for transformations (see 3.3.7).

If the Segment Open State is NO when the COPY SEGMENT function is interpreted, then the copied objects are treated as non-retained data and the associated transformations are applied during rendering. If the Segment Open State is YES when COPY SEGMENT is interpreted, the group of objects from the source segment with their associated copy and (optional) source segment transformations are added to the open segment. The Segment Transformation attribute value of the open segment also affects the eventual rendering of the group of objects copied. Transformations associated during a copy segment operation are object attribute values. They do not affect the open segment, only the group of objects copied. These transformations are never dissociated during subsequent copy segment operations regardless of parameter values or inheritance filter settings. The replacement, transformation concatenation, and intersection of clip rectangles according to the setting of the inheritance filter is discussed in 5.2.5.

The client specifies a copy transformation parameter with the invocation of COPY SEGMENT. The copy transformation is associated with the graphic objects as they are copied and it affects any associated clip rectangles. The transformation may translate, scale, rotate, and skew, though some implementations may not support rotation or skewing of clip rectangles – whether this is supported or not is indicated by the Transform Clip Region Effectiveness entry in the Segment Description Table.

The client also specifies whether the source segment's Segment Transformation attribute value is also to be concatenated with the copied objects' associated transformation. If the source segment's Segment Transformation attribute value is concatenated, then the resulting association is such that when the transformations are subsequently applied, the segment transformation is applied before the copy transformation. In general, they are not commutative.

NOTE – Repeated copying may occur, in which case, multiple pairs of source segment and copy transformations may be applied to the objects. These will be applied in the order from the first copy through to the last copy.

Examples of the use of the COPY SEGMENT and INHERITANCE FILTER functions are given in annex D.