INTERNATIONAL STANDARD

ISO/IEC 9636-6

First edition
1991-12-15

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

Part 6:
Raster

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

Partie 6: Raster

Reference number
ISO/IEC 9636-6:1991(E)

Contents

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

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 9636-6 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology.*

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

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

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

Introduction

This part of ISO/IEC 9636 describes the functions of the Computer Graphics Interface concerned with raster graphic specific devices.

The functional capability incorporated in this part of ISO/IEC 9636 is concerned with creating, manipulating, displaying and retrieving information stored as pixel data below the CGI in a device independent, yet efficient manner.

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

This page intentionally left blank

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

Part 6: Raster

1 Scope

This part of ISO/IEC 9636 describes those functions of the Computer Graphics Interface concerned with creating, modifying, retrieving, and displaying portions of an image stored as pixel data. It includes functionality for combining such images.

This part of ISO/IEC 9636 is part 6 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 (see ISO/IEC 9636-1, 5.2.1 and figures 6 and 7) and in clause 4.

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

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/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 9636-4 : 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 9637-1 : -[1] *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding — Part 1: Character encoding.*

ISO/IEC 9637-2 : -[1] *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding — Part 2: Binary encoding.*

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

1) To be published.

3 Raster concepts

3.1 Introduction

This part of ISO/IEC 9636 defines a set of functions for creating, modifying, retrieving, and displaying information stored as pixel data below the CGI. This functionality is divided into the following areas

- *Raster control functions,* including functions for the creation and deletion of bitmaps, and the selection of drawing and display bitmaps, and the control of raster transparency and mapped bitmap expansion.
- *Raster attribute functions,* for setting particular attributes which have significance with other graphical output, defined in ISO/IEC 9636-3 and this part of ISO/IEC 9636, when used in conjunction with raster functionality.
- *Raster operation functions,* including display and retrieval of pixel array data, and various forms of bitmap manipulation operations (bitblts) including movement, combination, and replication of bitmap regions.
- *Raster inquiry functions,* which provide access to the description tables and state lists defined in this part of ISO/IEC 9636.

3.2 Architectural concepts

3.2.1 Bitmaps

A bitmap is a region of computer memory that can be treated as if it were a rectangular array of pixels. Bitmaps are created and maintained below the CGI, and their internal format is hidden from the CGI client. Bitmaps never share common memory. Functions are provided to allow a CGI client to create and manage bitmaps.

Bitmaps may be INDEXED, DIRECT, and MIXED, as specified by the Array of Supported Bitmap Mode Combinations entry in the Raster Description Table, indicating the type of colour values that may be contained in a bitmap. When MIXED, both indexed and direct colour values may exist simultaneously in a bitmap.

3.2.2 Displayable bitmaps

Displayable bitmaps are special bitmaps that can be displayed on the display surface. The client can select which displayable bitmap is to be displayed on the display surface and a different displayable bitmap may be selected by the client at any time. There are from 1 to N predefined displayable bitmaps, where N is defined in the Raster Description Table. Predefined displayable bitmaps are all created to be the same size as the display surface and may not be deleted. Additional displayable bitmaps may be created at any arbitrary rectangular size by the client. Client created bitmaps may be deleted.

In some environments the displayed bitmap is subject to spontaneous change in dimensions; for example, in window-managed environments. Whether or not such spontaneous change in the displayable bitmap dimensions is allowed is indicated by an entry in the Output Device Description Table defined in ISO/IEC 9636-2. If such spontaneous change in dimensions is allowed, the device coordinate information in the Bitmap State List for the currently displayed bitmap will be modified to reflect the change. Thus, to detect such a spontaneous change, periodic polling by the client of this information is necessary. Alternatively, the CGI client may receive asynchronous notification of a change of dimensions by the environment via a non-CGI interface.

When the currently selected display bitmap does not completely cover the display surface, the CGI allows latitude about whether the contents of previous displayable bitmaps are visible in those regions of the display surface not covered by the currently selected display bitmap. The Previous Display Bitmap Data entry in the Raster Description Table specifies the implemented behaviour, which may be either CLEARED or PRESERVED. A value of PRESERVED indicates that it is possible for the display surface to be dirty even though the drawing surface is clear. If the contents of a previous displayable bitmap are visible and the bitmap is selected as the current drawing bitmap, it is implementation-dependent whether changes to the bitmap are visible.

3.2.3 Non-displayable bitmaps

These bitmaps cannot be displayed directly, but the information within a non-displayable bitmap may be moved to or combined with a displayable bitmap.

Pixels in a non-displayable bitmap are treated as if their aspect ratio is the same as that of pixels in a displayable bitmap for the device.

Non-displayable bitmaps may be one of two types: FULL DEPTH or MAPPED, as illustrated in figure 1.

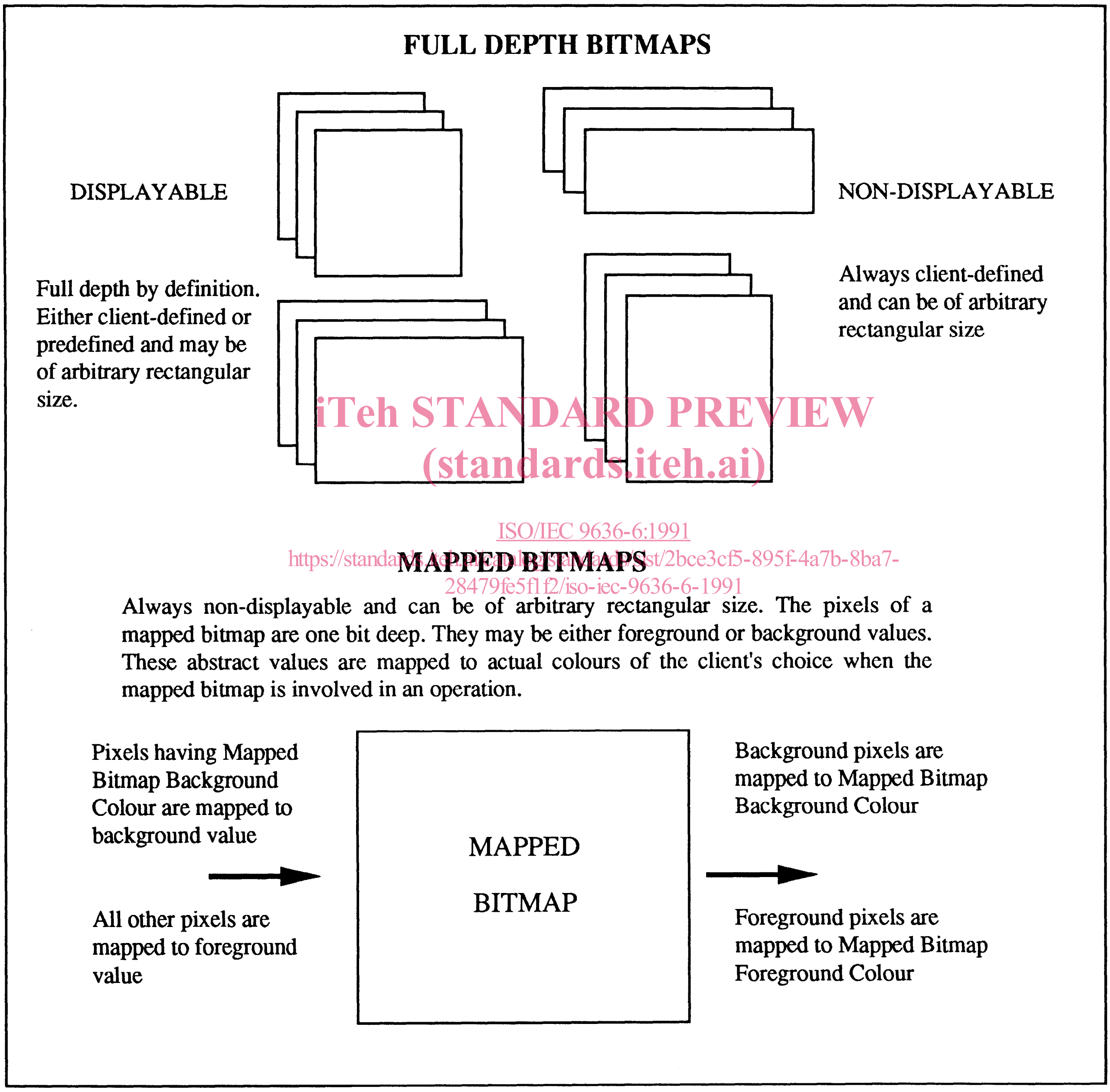


Figure 1 – Types of bitmaps

The pixels of full-depth bitmaps have the same number of bits per pixel as displayable bitmaps. Thus, a full-depth bitmap has the same colour capabilities as the physical device, and maintains the pixel values as CI or CD colour specifiers in the same manner as displayable bitmaps.

The pixels of a mapped bitmap can assume only the two abstract values *foreground* or *background*, which are mapped to actual colours of the client's choice when the bitmap is involved in an operation. Mapped bitmaps are thus convenient for such purposes as storing bitmap character fonts.

The effect of PREPARE DRAWING SURFACE on a mapped bitmap is to set all pixels of the mapped bitmap to the background value.

3.2.4 Bitmap identifiers

Bitmaps are referenced through an identifier. The first N consecutive identifiers identify the N predefined displayable bitmaps. The value of N is contained in the Raster Description Table and is an invariant resource of the CGI Virtual Device. The list of predefined displayable bitmap identifiers is given by the default value of the List Of Bitmap Identifiers entry (for displayable bitmaps) in the Raster State List.

The client can define its own identifiers for bitmaps it creates or it may have identifiers defined for it by using the function GET NEW BITMAP IDENTIFIER. The identifier is passed as an input parameter to the CREATE BITMAP function which then creates a bitmap with the given identifier.

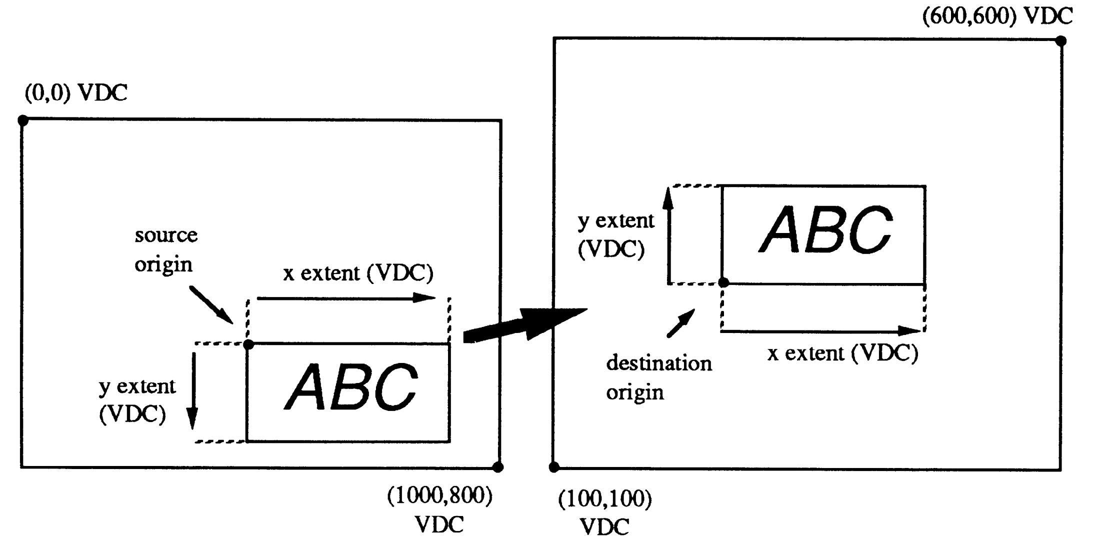
3.3 Control of bitmap manipulations

3.3.1 The drawing bitmap

The client can select which bitmap is affected by drawing operations. This is referred to as the *drawing bitmap*. Any bitmap may be selected as the drawing bitmap regardless of its depth type or displayability: FULL DEPTH or MAPPED, NON DISPLAYABLE or DISPLAYABLE. It is the currently selected drawing bitmap, not the current display bitmap, that is affected by functions such as PREPARE DRAWING SURFACE, VDC EXTENT, DEVICE VIEWPORT, POLYLINE, DRAW ALL SEGMENTS. The client should select explicitly the current displayed bitmap as the drawing bitmap if such functions are to apply to the displayed bitmap.

3.3.2 Two-operand bitblts

A two-operand bitblt function is provided to support operations that move and combine the contents of rectangular regions of bitmaps in memory. The regions of interest are specified using points in VDC. The actual movement of data takes place without regard to the VDC coordinate systems (see figure 2). If the destination for these operations is also the currently selected display bitmap then these operations may affect the displayed picture. The two-operand bitblt function combines the source and destination pixels according to the bitmap type and drawing modes in tables 1 and 2. The particular drawing mode used is specified as one of the parameters of the two-operand bitblt function.



Note, the actual transfer of pixels between source and destination regions is performed without regard to the coordinate system. The upper left source pixel shall always be moved to the upper left destination pixel and is independent of the two different VDC systems of the bitmaps.

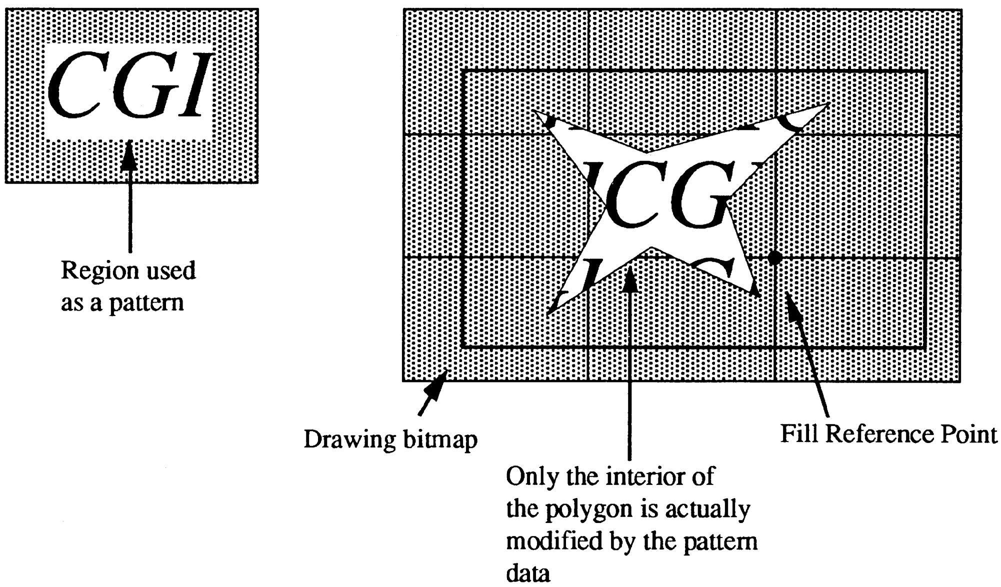
Figure 2 – Bitblt regions

3.3.3 Tile three-operand bitblt

Tile three-operand bitblts support the combination of two sources and a destination for the data movement. One of the sources may be used as a replicating tile and is referred to as the pattern. The drawing mode-3 operation used is specified as one of the parameters of the tile three-operand bitblt function, (refer to 5.4.5 and annex D). Annex E.3 illustrates the use of a tile three-operand bitblt in tiling a filled area (the letter P) in the destination bitmap corresponding to the filled area given by the source bitmap.

3.3.4 Bitmaps regions used as patterns

In addition to using bitmap regions as tiles for a tile three-operand bitblt, they may also be used to provide the pattern data for fill objects defined in ISO/IEC 9636-3. Figure 3 illustrates the use of a bitmap region as a pattern for filling.



When a bitmap region is used as a pattern, the corner of the pattern region is placed at the fill reference point and the region is conceptually replicated in X and Y to completely fill the destination space. Only the area of the bitmap which is being filled (e.g. the polygon) will actually be modified by the pattern data.

Figure 3 – Bitmap regions used as patterns

There is no restriction on the size of a bitmap region that may be used as a pattern. By using a bitmap region as large as the area being filled and properly aligning it by setting the Fill Reference Point, the pattern bitmap region may be used to achieve the effect of a stencil, rather than a tile. Bitmap regions include their boundary.

The default Fill Bitmap is a mapped bitmap of one pixel having the Mapped Bitmap Foreground Colour.

3.3.5 Drawing modes

Drawing modes are used to select the way in which pixels are combined during rendering or bitblt operations. In particular, it applies to the rendering of graphic objects (defined in ISO/IEC 9636-3) into the drawing bitmap. In this case, the drawing mode used to determine the combining operation is the object's DRAWING MODE attribute value. During object creation, this attribute value is taken from the Drawing Mode entry in the Raster State List. The drawing mode concept also applies to raster operation functions where the specification of a particular drawing mode is supplied as a separate parameter.

A device-dependent colour index value may be used if the result of a combining operation using indexed colour values is undefined. The RGB components of direct colour values are combined separately. A device-dependent colour value may be used if the result of a combining operation on any of the individual RGB components results in an overflow.

Implementation-dependent colour value conversions may occur when using the raster operation functions if the pixel values of the source and destination bitmaps are not all specified as indexed colour values nor all specified as direct colour values. An implementation-dependent colour value may be placed in the drawing bitmap if the colour selection mode of the colour value resulting from a combining operation differs from the Bitmap Mode of the drawing bitmap.

3.3.6 Transparency

The concept of transparency is also applicable to raster operation functions. For these operations, the transparency value is supplied as a parameter of the raster operation functions. If the value of the transparency parameter is OPAQUE, all pixels transferred to the destination bitmap region will affect the destination. If the value of the transparency parameter is TRANSPARENT, only those pixels that either have, or expand to (in the case of mapped bitmaps), a value that is not that of the Transparent Colour entry in the Raster State List will affect the destination. (See also ISO/IEC 9636-3, 3.4.1.5.)

3.3.7 Raster operation functions with mapped bitmaps

In performing a raster operation function which involves a combining operation over pixel values from one or more mapped bitmaps, then, prior to the pixel combining operation, pixels in mapped bitmaps which have "foreground value" are expanded to the value of Mapped Bitmap Foreground Colour in the Raster State List and those which have "background value" are expanded to the value of Mapped Bitmap Background Colour in the Raster State List. Such expansion also occurs whenever the destination bitmap is full-depth, even when there is no effective combining operation (e.g. $d' <- s$).

In performing a raster operation function in which the destination bitmap is a mapped bitmap, then, after any pixel combining operations, destination pixels are set to "background value" whenever the result of the combining operation was a pixel with value equal to that of Mapped Bitmap Background Colour and to "foreground value" otherwise.

3.3.8 Rendering in full-depth bitmaps

The associated DRAWING MODE attribute value of a graphic object affects the way that object is rendered in the drawing bitmap. For each pixel affected in rendering an object, the drawing colour may either be the colour attribute value associated with the object (or edge) or the AUXILIARY COLOUR attribute value (if the associated TRANSPARENCY attribute is OPAQUE). The drawing colour is combined with the destination pixel in accordance with the associated DRAWING MODE attribute value and the result replaces the former value of the destination pixel. The Transparent Colour entry in the Raster State List has no affect on this rendering operation.

3.3.9 Rendering in mapped bitmaps

The rendering of an object in the drawing bitmap when that bitmap is a mapped bitmap, is affected by the Mapped Bitmap Background Colour and Mapped Bitmap Foreground Colour in the Raster State List. Conceptually, an affected destination pixel is expanded to a full-depth value according to whether it is foreground or background. It is then combined with the drawing colour (determined by the object's colour and transparency attributes) in accordance with the associated DRAWING MODE attribute value. If the result is equal to the value of Mapped Bitmap Background Colour then the affected destination pixel is set to the "background value", otherwise it is set to the "foreground value".

3.4 Pixel array

The PIXEL ARRAY function is related to the Bitblt's described above, in that it is not considered to create a graphic object and is treated in a manner similar to a Bitblt where the source array is provided by the client and the destination is the drawing bitmap. The colour information in the pixel array maps directly to the pixels of the destination bitmap. Thus the starting point and colour information are specified in a device-independent fashion, but the appearance of the final image depends directly on the resolution and aspect ratio of the target device. An MxN array of device-independent colour specifiers are assigned to an MxN array of pixels (assuming the x and y scale parameters are both 1). The differences between CELL ARRAY, PIXEL ARRAY, and Bitblt are illustrated in figure 4.

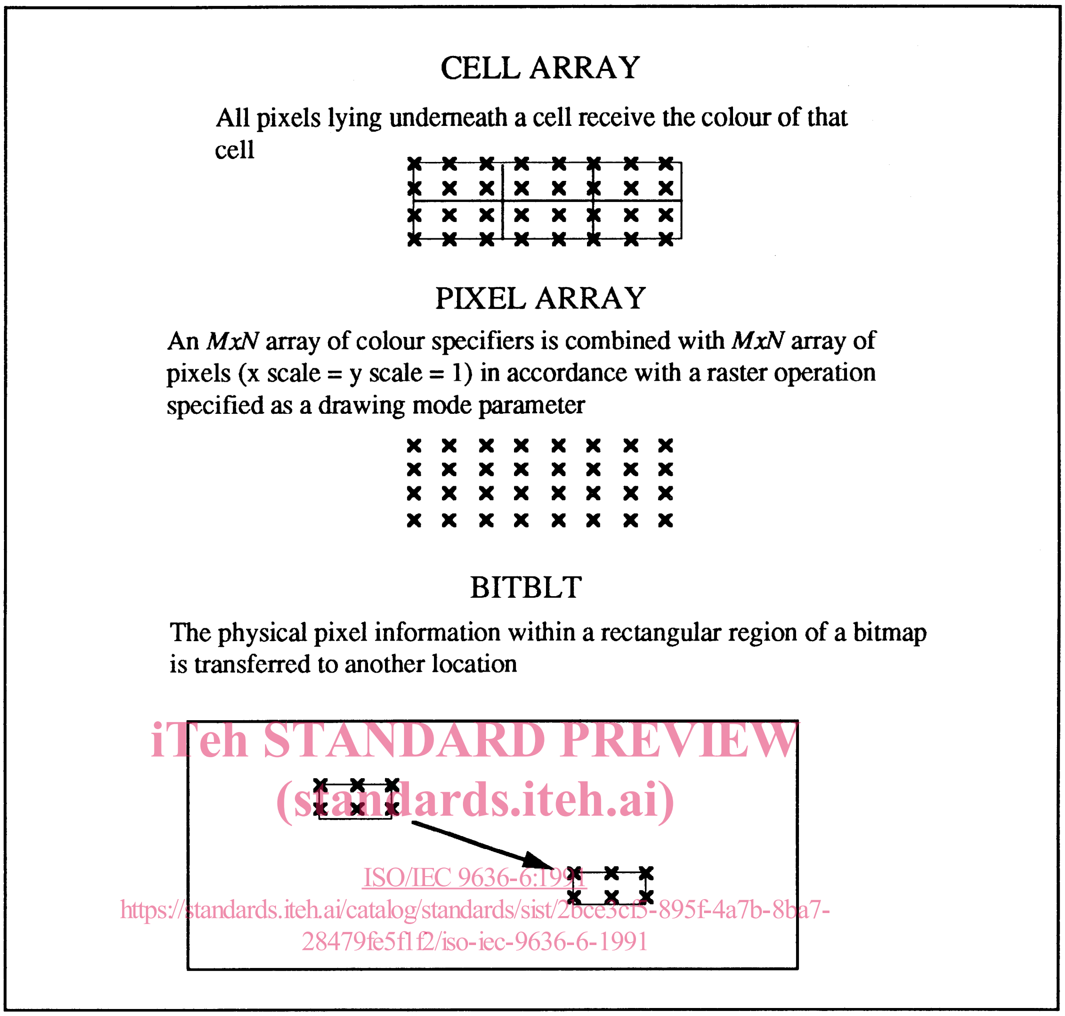


Figure 4 – A comparison of CELL ARRAY, PIXEL ARRAY, and Bitblt

The GET PIXEL ARRAY function returns a rectangular array of colour values from the identified source bitmap to the client.

3.5 The VDC-to-Device Mapping and clipping

3.5.1 Determining the position and size of created bitmaps

The position on the display surface and size, in pixels, of the bitmap to be created are derived from the VDC coordinates of the bitmap extent (specified as a parameter of the function CREATE BITMAP) transformed by the VDC-to-Device Mapping of the current drawing bitmap (see figure 5). In essence, the size, in pixels, is determined by passing the corner points of the bitmap extent through the VDC-to-Device Mapping. The x and y displacements from the first DC point to the second give the dimensions in pixels of the bitmap that the client desires to have created.

While the bitmap dimensions of an existing bitmap cannot be modified by the client, the client may modify the VDC-to-Device Mapping for the bitmap by selecting the bitmap as the drawing bitmap and then invoking the functions VDC EXTENT and DEVICE VIEWPORT (see ISO/IEC 9636-2).

Once created, the VDC-to-Device Mappings of different bitmaps are completely independent. The client might use the VDC EXTENT function to specify several different bitmaps with the same VDC Extents (even if the dimensions in pixels of the bitmaps were different) or might specify different VDC Extents for bitmaps having the same bitmap dimensions.