Starbase Reference

HP 9000 Series 300/800 Computers

HP Part Number 98592-90065

Hewlett-Packard Company
3404 East Harmony Road, Fort Collins, Colorado 80525

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MANUAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

WARRANTY

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Copyright © Hewlett-Packard Company 1989

This document contains information which is protected by copyright. All rights are reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

Restricted Rights Legend

Use, duplication or disclosure by the U.S. Government Department of Defense is subject to restrictions as set forth in paragraph (b)(3)(ii) of the Rights in Technical Data and Software clause in FAR 52.227-7013.

Use of this manual and flexible disc(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs can be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Copyright © AT&T, Inc. 1980, 1984

Copyright © The Regents of the University of California 1979, 1980, 1983

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

September 1989...Edition 1. This edition replaces part number 98592-90063.

Table of Contents

Section 1: User Commands

Section 3: Subroutine Libraries

Entry Name(Section) *name* **Description**

Entry Name(Section) *name* **Description**

Entry Name(Section) *name* **Description**

Entry Name(Section) *name* **Description**

Entry Name(Section) *name* **Description**

Section 4: File formats

NAME

pcltrans – translate a Starbase bitmap file into PCL raster graphics format.

SYNOPSIS

pcltrans [options] <file>

DESCRIPTION

The command *pcltrans* translates bitmap (raster graphics) data from a file (a starbase formatted bitmap file) into Printer Command Language (PCL) raster graphics format.

If input is redirected through a pipe a temporary file will be created for use by the internal pcltrans formatter.

Note that output is a series of PCL commands which consist of binary data and escape sequences. Since output is to standard out, pcltrans should be used as a filter with output redirected to a file or to some other process such as the lp spooler.

Unless modified by options, the output includes PCL commands to set printer resolution, begin raster graphics mode, transfer raster graphics by rows, end raster graphics mode, and the final byte of output is a form feed. RGB pixels are converted to shades of gray based on their intensity values. If pixels are not expanded, the output is left to right across the long dimension of the paper, and resolution is 300 dots per inch.

When PCL imaging functionality is invoked (via the **–I** option), a different set of defaults are activated. If the output device has color capabilities, the data is left in rgb format and the output device prints the output in color. The default render algorithm is ordered dither with a gamma correction value of 0.8. The output device renders the image at its highest possible resolution, and the output image is scaled for the best fit on the size of media loaded into the output device. The output is from left to right across the long dimension of the paper.

WARNING: If a PCL file that contains imaging commands is sent to a PCL device that does not understand this functionality, the output will be incorrect.

The following options are recognized by *pcltrans*:

Imaging Option:

–I PCL imaging capabilities are invoked. The output image is scaled for the best fit on the size of media loaded in the device. The image is rendered using ordered dither and a gamma correction value of 0.8. If the output device has color capabilities, the image is in color by default.

Rendering Options:

–a*alg* Render algorithm desired. Valid only in conjunction with the **–I** option. The possible values of *alg* are:

- 0 = no algorithm applied
- 1 = snap to printer PRIMARIES (same as the **–c–C** option)
- 2 = snap black to white and all other colors to black (can use for text screen dumps)
- 3 = color ordered dither (default)
- 4 = color error diffusion (same as the **–C** option)
- 5 = monochrome ordered dither
- 6 = monochrome error diffusion

–c Each pixel with a nonzero RGB value is converted to black. Default is conversion to the nearest available shade of gray. This option has a different effect if the **–C** option is also used (yields PRIMARY). The **–a** option overrides this option.

−C Each pixel is converted to the appropriate output color value using an error diffusion algorithm. This option also changes the effect of the **−c** option. Not all PCL printers support color mode. The **−a** option overrides this option.

−N Each pixel is converted to the appropriate output color value using an error diffusion algorithm with random noise. This option overrides the effect of the **−c** and **−C** options. Not all PCL printers support this color mode. This option is invalid with the **−I** option.

−c−C Each pixel is converted to the nearest PRIMARY color (red, green, blue, cyan, magenta, yellow, white, or black).

Positioning Options:

−R Output print orientation is left to right across the width of the paper (analogous to portrait mode on the Laserjet printer). Default is left to right across the length of the paper. Raster rotation is done in pcltrans rather than changing printer modes.

−s Raster graphics is started at the current printer cursor position. The first command in the output is Ec*t#R (set graphics resolution) followed by Ec*r1A (start raster graphics). Next follows individual rows of raster data prefixed with Ec*b#W[raster] (transfer raster graphics), and the last command in the output is Ec*rB (terminate raster graphics).

−x*inches*
Destination image offset x inches from the left boundary. Valid only in conjunction with the **−I** option.

−y*inches*
Destination image offset y inches from the top boundary. Valid only in conjunction with the **−I** option.

Sizing Options:

−I*length*
Length of the paper in inches. Default is 10.5 inches.

−w*width*
Width of paper in inches. Default is 8.0 inches.

−e*exp* Pixel expansion - indicates the expansion for each pixel in the bitmap and ranges from 1 to 8. For example, to expand from a single pixel to a 4x4 cell, the expansion parameter is set to 4. Default is 1.

−d*inches*
Width of the destination image in inches. Valid only in conjunction with the **−I** option.

−h*inches*
Height of the destination image in inches. Valid only in conjunction with the **−I** option.

Background/Foreground Options:

−k Don't print the background color. Default is to print the background color.

−b*bcol* Background color table index for printing a single plane. Default is 0. This option only has effect when printing a single plane as specified with the **−p** option.

−f*fcol* Foreground color table index for printing a single plane. Default is 1. This option only has effect when printing a single plane as specified with the **−p** option.

Image Reproduction Options:

−p*plane*
Bitmap plane to be printed when only a single plane is to be printed. Default is to print a full depth bitmap. If the plane specified in this option is a non-negative number and is not contained in the bitmap file, the pcltrans formatter will exit with error. If the

plane specified in this option is negative the formatter will default to all planes.

−T Print on transparency film. Valid only in conjunction with the **−I** option.

−z*opt* Scale algorithm to apply to the image, only if the destination image is smaller than the source in pixels. Valid only in conjunction with the **−I** option. The possible values of *opt* are:

0 = algorithm to apply to an image with a light background
1 = algorithm to apply to an image with a dark background

−g*value*

Gamma correction value to apply to the image. Valid values are between 0.0 and 2.0. Valid only in conjunction with the **−I** option.

−q Presentation mode quality. A higher quality available on the ink jet technology printers. Valid only in conjunction with the **−I** option.

−K*kmem*

Amount of memory in kilobytes to use in processing the image. With increased memory, the processing can be done much faster. If the user specifies a larger amount of memory than is available, the processing will actually by slower due to virtual memory activity. This is only valid in conjunction with the **−I** option.

EXAMPLES

A typical case is formatting a file for output at 150 dots per inch resolution to a Laserjet Plus printer. The desired plot orientation is left to right across the width of the page. The output in this example is then piped to lp in raw mode so that no processing of the data is done by lp.

```
pcltrans -r150 -R myfile | lp -oraw
```

Another typical example is to format a file to print with imaging capabilities invoked, color error diffusion, a gamma correction value of 0.6, and in presentation mode, piped to the lp spooler.

```
pcltrans -I -a4 -g0.6 -q myfile | lp -oraw
```

SEE ALSO

HP-UX Reference,lp(1M), bitmapfile(5).
Starbase Reference,bitmap_to_file(3G).
Starbase Device Drivers Library, "PCL Formatter", "PCL Imaging Formatter.
Starbase Graphics Techniques, "Storing and Printing Images".

NAME

screenpr − capture the screen raster information and translate into PCL raster graphics format.

SYNOPSIS

screenpr [options]

DESCRIPTION

The command *screenpr* captures screen raster data and translates the bitmap into Printer Command Language (PCL) raster graphics format. Output is a series of PCL commands which consist of binary data and escape sequences. Since output is to standard out, *screenpr* should be used as a filter with output redirected to a file or to some other process such as the lp spooler.

The output, unless modified, includes PCL commands to set printer resolution, begin raster graphics mode, transfer raster graphics by rows, end raster graphics mode, and output a form feed as the final byte. In addition, rgb pixels are converted to shades of gray based on their intensity values. If pixels are not expanded, the output is left to right across the long dimension of the paper, and resolution is 300 dots per inch.

When PCL imaging functionality is invoked (via the **−I** option) a different set of defaults are activated. The data is left in rgb format and if the output device has color capabilities, the output is printed in color. The default render algorithm is ordered dither with a gamma correction value of 0.8. The output device will render the image at its highest possible resolution, and the output image is scaled for the best fit on whatever size media is loaded into the output device. The output is left to right across the long dimension of the paper.

Warning: If a PCL file that contains imaging commands is sent to a PCL device that does not understand this functionality, the output will be incorrect.

The following options are recognized by *screenpr*:

Imaging Option:

−I PCL imaging capabilities are invoked. The output image is scaled for the best fit on the size of media loaded in the device. The image is rendered using ordered dither and a gamma correction value of 0.8. If the output device has color capabilities, the image is in color by default.

Rendering Options:

−a*alg* Render algorithm desired. Valid only in conjunction with the **−I** option. The possible values of *alg* are:

0 = no algorithm applied
1 = snap to printer PRIMARIES (same as the **−c−C** option)
2 = snap black to white and all other colors to black (can use for text screen dumps)
3 = color ordered dither (default)
4 = color error diffusion (same as the **−C** option)
5 = monochrome ordered dither
6 = monochrome error diffusion

−c Each pixel with a nonzero RGB value is converted to black. Default is conversion to the nearest available shade of gray. This option has a different effect if the **−C** option is also used (yields PRIMARY). The **−a** option overrides this option.

−C Each pixel is converted to the appropriate output color value using an error diffusion algorithm. This option also changes the effect of the **−c** option. Not all PCL printers support color mode. The **−a** option overrides this option.

−N Each pixel is converted to the appropriate output color value using an error diffusion algorithm with random noise. This option overrides the effect of the **−c** and **−C**

options. Not all PCL printers support this color mode. This option is invalid with the −**I** option.

−**c**−**C** Each pixel is converted to the nearest PRIMARY color (red, green, blue, cyan, magenta, yellow, white, or black).

Positioning Options:

−**R** Output print orientation is left to right across the width of the paper (analogous to portrait mode on the Laserjet printer). Default is left to right across the length of the paper. Raster rotation is done in screenpr rather than changing printer modes.

−**s** Raster graphics is started at the current printer cursor position. The first command in the output is Ec*t#R (set graphics resolution) followed by Ec*r1A (start raster graphics). Next follows individual rows of raster data prefixed with Ec*b#W[raster] (transfer raster graphics), and the last command in the output is Ec*rB (terminate raster graphics).

−**x***inches*

Destination image offset x inches from the left boundary. Valid only in conjunction with the −**I** option.

−**y***inches*

Destination image offset y inches from the top boundary. Valid only in conjunction with the −**I** option.

Sizing Options:

−**I***length*

Length of the paper in inches. Default is 10.5 inches.

−**w***width*

Width of paper in inches. Default is 8.0 inches.

−**e***exp* Pixel expansion - indicates the expansion for each pixel in the bitmap and ranges from 1 to 8. For example, to expand from a single pixel to a 4x4 cell, the expansion parameter is set to 4. Default is 1.

−**d***inches*

Width of the destination image in inches. Valid only in conjunction with the −**I** option.

−**h***inches*

Height of the destination image in inches. Valid only in conjunction with the −**I** option.

Background/Foreground Options:

−**k** Don't print the background color. Default is to print the background color.

−**b***bcol* Background color table index for printing a single plane. Default is 0. This option only has effect when printing a single plane as specified with the −**p** option.

−**f***fcol* Foreground color table index for printing a single plane. Default is 1. This option only has effect when printing a single plane as specified with the −**p** option.

Image Reproduction Options:

−**p***plane*

Bitmap plane to be printed when only a single plane is to be printed. Default is to print a full depth bitmap. If the plane specified in this option is a non-negative number and is not contained in the bitmap file, the pcltrans formatter will exit with error. If the plane specified in this option is negative the formatter will default to all planes.

−**T** Print on transparency film. Valid only in conjunction with the −**I** option.

−**z***opt* Scale algorithm to apply to the image, only if the destination image is smaller than the source in pixels. Valid only in conjunction with the −**I** option. The possible values of

opt are:
0 = algorithm to apply to an image with a light background
1 = algorithm to apply to an image with a dark background

−g*value*
Gamma correction value to apply to the image. Valid values are between 0.0 and 2.0. Valid only in conjunction with the **−I** option.

−q Presentation mode quality. A higher quality available on the ink jet technology printers. Valid only in conjunction with the **−I** option.

Image Information:

−F*devfile*
If the graphics monitor's default is changed or multiple monitors are present, the user can specify the device file in which the bitmap is displayed. Default is **/dev/crt**.

−D*pixels*
Width of the original image in pixels. Default is the entire screen.

−H*pixels*
Height of the original image in pixels. Default is the entire screen.

−X*pixels*
Offset of the source image from the screen's left boundary.

−Y*pixels*
Offset fo the source image from the screen's top boundary.

EXAMPLES

A typical case is formatting a file for output at 150 dots per inch resolution to a Laserjet Plus printer. The desired plot orientation is left to right across the width of the page. The output in this example is then piped to lp in raw mode so that no processing of the data is done by lp.

```
screenpr -r150 -R myfile | lp -oraw
```

Another typical example is to format a file to print with imaging capabilities invoked, color error diffusion, a gamma correction value of 0.6, and to obtain the bitmap from a device file **/dev/other** and pipe to the lp spooler.

```
screenpr -I -a4 -g0.6 -q -F/dev/other | lp -oraw
```

SEE ALSO

*HP-UX Reference,*lp(1M), bitmapfile(5).
*Starbase Reference,*bitmap_to_file(3G).
Starbase Device Drivers Library, "PCL Formatter", "PCL Imaging Formatter".
Starbase Graphics Techniques, "Storing and Printing Images".

NAME

append_text – output a string of characters.

SYNOPSIS

C Syntax:

```
void append_text(fildes,string,xform,more);
int fildes,xform,more;
char *string;
```

FORTRAN77 Syntax:

```
subroutine append_text(fildes,string,xform,more)
integer*4 fildes,xform,more
character*(*) string
```

Pascal Syntax:

```
procedure append_text(fildes:integer;string:string255;xform,more:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

string ASCII string of characters terminated with a null character.

xform Enumerated value specifying the type of transformation to perform on the font coordinates:

VDC_TEXT = vdc
WORLD_COORDINATE_TEXT = world coordinate
TOS_TEXT = top of matrix stack

more Set to TRUE (1) if more characters are to be buffered with the current text and FALSE (0) if no buffering is needed.

Discussion

The *append_text* primitive draws characters starting at the current VDC or WC position. If **more** is non-zero, characters are buffered, awaiting more text before text alignment and drawing are performed. This allows changing of text color, current font, and other such calls between *append_text* and text calls. If **more** is zero, alignment and drawing are done immediately. **Xform** specifies the type of transformation to be applied to the font coordinates.

If **xform** equals VDC_TEXT (zero), the current font transformation (*character_height*, *character_width*, etc.) is pre-concatenated with the VDC to device coordinate transformation and used as the font-to-device coordinate transformation.

If **xform** equals WORLD_COORDINATE_TEXT (one), the current font transformation is pre-concatenated with the transformation matrix on top of the matrix stack and used as the font-to-device coordinate transformation.

If **xform** equals TOS_TEXT (two), the top of the matrix stack is used as the font-to-device coordinate transformation matrix.

SEE ALSO

character_height(3G), text(3G), text_alignment(3G), text_color(3G), text_font_index(3G), text_line_path(3G), text_line_space(3G), text_orientation(3G), text_path(3G). text_precision(3G), text_switching_mode(3G).

NAME

arc, intarc, intpartial_arc, partial_arc – define an elliptical or circular region to be filled and/or edged.

SYNOPSIS

C Syntax:

```
void arc(fildes,x_radius,y_radius,x_center,y_center,start,stop,rotation,
        close_type);
int fildes,close_type;
float x_radius,y_radius,x_center,y_center,start,stop,rotation;

void intarc(fildes,radius,x_center,y_center,start,stop,close_type);
int fildes,radius,x_center,y_center,close_type;
float start,stop;

void intpartial_arc(fildes,radius,x_center,y_center,start,stop,close_type,
        closure);
int fildes,radius,x_center,y_center,close_type,closure;
float start,stop;

void partial_arc(fildes,x_radius,y_radius,x_center,y_center,start,stop,
        rotation,close_type,closure);
int fildes,close_type,closure;
float x_radius,y_radius,x_center,y_center,start,stop,rotation;
```

FORTRAN77 Syntax:

```
subroutine arc(fildes,x_radius,y_radius,x_center,y_center,start,stop,
        rotation,close_type)
integer*4 fildes,close_type
real x_radius,y_radius,x_center,y_center,start,stop,rotation

subroutine intarc(fildes,radius,x_center,y_center,start,stop,close_type)
integer*4 fildes, radius,x_center,y_center,close_type
real start,stop

subroutine intpartial_arc(fildes,radius,x_center,y_center,start,stop,
        close_type,closure)
integer*4 fildes, radius,x_center,y_center,close_type,closure
real start,stop

subroutine partial_arc(fildes,x_radius,y_radius,x_center,y_center,
        start,stop,rotation,close_type,closure)
integer*4 fildes,close_type,closure
real x_radius,y_radius,x_center,y_center,start,stop,rotation
```

Pascal Syntax:

```
procedure arc(fildes:integer;x_radius,y_radius,x_center,y_center,start,
    stop,rotation:real;close_type:integer);

procedure intarc(fildes,radius,x_center,y_center:integer;start,stop:real,
    close_type:integer);

procedure intpartial_arc(fildes,radius,x_center,y_center:integer;
    start,stop:real;close_type,closure:integer);

procedure partial_arc (fildes:integer; x_radius,y_radius,x_center,y_center,
    start,stop,rotation:real;close_type, closure:integer);
```

DESCRIPTION

Input Parameters

fildes	Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.
radius	Radius of the arc.
x_radius,y_radius	Radii of the arc in the *x* and *y* directions respectively.
x_center,y_center	Coordinate of the arc center.
start,stop	Specify the limits of an elliptical arc or circular arc in radians.
rotation	Specifies the amount the arc is to be rotated in radians.
close_type	specifies how an arc is to be closed. Selected from: NO_CHORD (0), PIE (1) or CHORD (2).

Discussion

An elliptical arc or circular arc of specified radii centered at **(x_center,y_center)** is filled and/or outlined according to the current interior style. The arc is closed in one of three ways based on **close_type**. If **close_type** is NO_CHORD, only the arc is drawn. If **close_type** is PIE, two edges are added from the center of the arc to both the **start** and **stop** position. If **close_type** is CHORD, one edge is added that connects the **start** and **stop** postion of the arc. As with all output primitives, the arc is affected by the current drawing mode and write enable. The curve is drawn with chords of length specified in *curve_resolution*. For *partial_arc* and *intpartial_arc*, the vertices of the arc are added to the polygon vertex list, as in the case of *partial_polygons*. Arcs are drawn from **start** to **stop**. If **stop** is greater than **start**, the arc is drawn in a counterclockwise direction. If **stop** is less than **start**, the arc is drawn in a clockwise direction.

If the transform mode has been set to THREE_D, the *z* value used is that of the current position. Circles and circular arcs are drawn by specifying **x_radius** and **y_radius** to be equal.

Non-filled arcs are generated by setting the interior style to INT_HOLLOW and edged.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

NOTE

Doing any other Starbase call besides *intpartial_arc, intpartial_circle, intpartial_polygon, partial_arc, partial_ellipse, partial_polygon2d, partial_polygon3d*, or *dcpartial_polygon*, in the middle of a list of procedure calls that add vertices to the polygon buffer and before a call to *arc, polygon, rectangle*, or *ellipse* produces unpredictable, device-dependent results.

When using a distorted mapping (x units do not equal y units), integer arcs may not remain circular. This effect is unpredictable and device-dependent, and may not remain consistent with future releases.

In some cases, arcs may actually be rendered as polygons. Certain devices have a limit on the number of polygon vertices they support (see the *Starbase Device Drivers Manual* section for your particular device). If this limit is exceeded, you will receive a warning. To eliminate this warning, adjust the step size using the *curve_resolution* command.

SEE ALSO

circle(3g), curve_resolution(3G), drawing_mode(3G), ellipse (3G), fill_color(3G), interior_style(3G),
perimeter_color(3G), perimeter_type(3G), perimeter_repeat_length(3G), polygon(3G),
Starbase Graphics Techniques.

NAME

await_event – wait for event to occur and be buffered then return class of the event (LOCATOR or CHOICE)

SYNOPSIS

C Syntax:

```
void await_event(await_fid,timeout,valid,class);
int await_fid;
float timeout;
int *valid,*class;
```

FORTRAN77 Syntax:

```
subroutine await_event(await_fid,timeout,valid,class)
integer*4 await_fid
real timeout
integer*4 valid,class
```

Pascal Syntax:

```
procedure await_event(await_fid:integer;timeout:real;var valid,
        class:integer);
```

DESCRIPTION

Input Parameters

await_fid Specifies how *await_event* waits. If **await_fid** is the file descriptor of an opened graphics input device, wait for events from that device. If **await_fid** is −1 (minus one), wait for events from any device.

timeout Maximum time, in seconds, to wait for an event. **Valid** is set to FALSE (0) if time runs out.

Output Parameters

valid Set to TRUE (1) if the event data is valid;
FALSE (0) if the event data is invalid.

class May be one of the following classes: LOCATOR or CHOICE. The integer values assigned for these class designators are found in the Starbase **include** file used with your program.

Discussion

If **await_fid** is minus one (−1), *await_event* waits for an event from any device. If *await_fid* is a valid file descriptor, *await_event* waits for an event from the corresponding device. If an event from a suitable device is already in the queue, then this call returns immediately with the first such event. If no such event is in the queue, *await_event* waits for an event from an appropriate device to be placed in the event queue. This is a timed wait with the interval specified by *timeout*. If no event occurs before time runs out, **valid** is set to FALSE (0), and **class** is undefined. If an event is found before time runs out, **class** is set to the event class (CHOICE or LOCATOR) and **valid** is set to TRUE.

This routine does not read and remove the event from the queue. It determines the existence of an event and the class of input involved. Once *await_event* has indicated a valid event, either *read_locator_event* or *read_choice_event* should be used to read the event and remove it from the event queue.

The procedures *enable_events* and *disable_events* are used to enable and disable the event generation capability of choice and locator devices.

If **await_fid** is an invalid file descriptor, a *device not initialized* error is reported and the procedure is ignored.

There is one common event queue for all physical devices. All events are entered in the order in which they occur. This allows time ordering to be established between several input devices. Simultaneous events are placed in the queue in an implementation-dependent order within their group, with all but the last one indicating **simultaneous_event_follows** (see **message_link** parameter used in *read_locator_event* and *read_choice_event* procedures).

Valid information may be read from the queue by using the *read_locator_event* or *read_choice_event* procedures.

Since TRUE and FALSE are reserved words in Pascal, integer values 1 and 0 are used instead when calling from Pascal.

SEE ALSO

disable_events(3G), enable_events(3G), read_locator_event(3G), read_choice_event(3G).

NAME

await_retrace – wait for vertical retrace on raster scanning devices.

SYNOPSIS

C Syntax:

```
void await_retrace(fildes);
int fildes;
```

FORTRAN77 Syntax:

```
subroutine await_retrace(fildes)
integer*4 fildes
```

Pascal Syntax:

```
procedure await_retrace(fildes:integer);
```

DESCRIPTION

INPUT PARAMETER

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

Discussion

Await_retrace waits in a busy loop until vertical blanking begins on raster scan CRT devices. Vertical blanking is the part of the retrace cycle when the electron beam is turned off and moved from the lower-right corner of the display to the upper-left corner. Vertical blanking time is device dependent, ranging from 1 ms to 0.1 ms on current devices. During vertical blanking, any needed changes to the frame buffer, control registers, or color map (such as when double-buffering) do not appear on the CRT, and thus do not produce any flicker and/or picture distortion.

It should also be noted that the *await_retrace* procedure may wait up to one full retrace cycle, (1/60 second on most displays). This procedure should only be used when making changes that can be completed in a very short time and which are not done more often than 1/60th of a second, such as when performing double buffering or cursor dragging. This procedure should not be used between primitives when drawing a picture because it can slow processing speeds considerably with no visible improvement in the display.

The *display_enable* command calls *await_retrace* before changing its state.

Some devices may not support this procedure.

SEE ALSO

display_enable(3G), *Starbase Device Drivers Library.*

NAME

backface_control – define aspects of backfacing polygons.

SYNOPSIS

C Syntax:

```
void backface_control(fildes,rev_normals,color,red,green,blue);
int fildes,rev_normals,color;
float red,green,blue;
```

FORTRAN77 Syntax:

```
subroutine backface_control(fildes,rev_normals,color,red,green,blue)
integer*4 fildes,rev_normals,color
real red,green,blue
```

Pascal Syntax:

```
procedure backface_control(fildes,rev_normals,color:integer;
                           red,green,blue:real);
```

DESCRIPTION

Input Parameters

fildes	Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.
rev_normals	If TRUE (1), any normals associated with polygons point in either direction when light source equations are applied.
color	If TRUE (1), backfacing polygons are rendered with the backface polygon color.
red, green, blue	Values (in the range 0.0 to 1.0) that define the backface polygon color.

Discussion

The **rev_normals** flag is useful in the case where normals are being calculated on polygons whose vertices are not arranged in a consistently clockwise or counter-clockwise order (see *vertex_format* procedure). It is also useful for polygons that are to be viewed as two-sided surfaces.

If **color** is TRUE, polygons that are backfacing with respect to the viewpoint are rendered by the light source equations using polygon or vertex normals pointing in the opposite direction and a new surface color defined by **red,green,blue**. Polygons can be thought of as two-sided surfaces where front-side surface characteristics are defined using *fill_color* and *surface_model* procedures and the back-side surface characteristics are defined using *backface_control*. If backface culling is enabled by the *hidden_surface*(3G) **cull** parameter, any subsequent call to *backface_control* with **color** TRUE has no effect because backface culling remains active. See "Exceptions to Standard Starbase Support" in the *HP98721 Device Driver* section of the *Starbase Device Drivers Library Manual.*

This routine, although still supported, has been replaced by *bf_control*. A separate set of attributes for backfacing polygons can be specified by calls to *bf_control* and related routines listed under the SEE ALSO heading.

SEE ALSO

bf_control(3G), interior_style(3G), perimeter_color(3G), surface_model(3G), surface_coefficients(3G), fill_color (3G), hidden_surface(3G), vertex_format(3G), *Starbase Device Drivers Library Manual.*

NAME

background_color – set background color by color table index or color value for painting by *clear_view_surface*

SYNOPSIS

C Syntax:

```
void background_color_index(fildes,index);
int fildes,index;

void background_color(fildes,red,green,blue);
int fildes;
float red,green,blue;
```

FORTRAN77 Syntax:

```
subroutine background_color_index(fildes,index)
integer*4 fildes,index

subroutine background_color(fildes,red,green,blue)
integer*4 fildes
real red,green,blue
```

Pascal Syntax:

```
procedure background_color_index(fildes,index:integer);

procedure background_color(fildes:integer;red,green,blue:real);
```

DESCRIPTION

INPUT PARAMETERS

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

index Integer index into the device color table. Background color is determined by the red, green, and blue values specified at this table location.

red,green,blue Specifies relative intensity of each color when composing the background color. The floating-point value for each color can range from 0 to 1 where 0 and 1 specify minimum and maximum limit, respectively. Monochrome devices determine intensity by using the equation:

$$I = 0.30(red) + 0.59(green) + 0.11(blue)$$

Discussion

When background color is set by using red, green, and blue parameters, determination of actual output depends on the state of the mode parameter set by *shade_mode*:

CMAP_NORMAL	Default mode if *shade_mode* has not been called. The color table is searched for an index pointing to the color in RGB space closest to the one specified.
CMAP_MONOTONIC	Color values are converted to an intensity determined by the equation: I=0.30(red)+0.59(green)+0.11(blue) This intensity is mapped to an index using the minimum and maximum defined by *shade_range*.
CMAP_FULL	Color values are mapped directly to a index with the assumption that the color map is set up with a predefined full color state.

If *clear_control* is set to **clear_all_banks**, the frame buffer is cleared to zero instead of the background color when *clear_view_surface* is called.

The background is repainted with the specified index by subsequent *clear_view_surface* commands. The area to be repainted is selected by *clear_control*, and can be:

Virtual device coordinate area
Clip rectangle area
Entire display area
Viewport area

If fildes specifies a hardcopy device, the background color cannot be changed.

DEFAULTS

Color index 0 (zero).

SEE ALSO

bank_switch(3G), clear_view_surface(3G), clear_control(3G), define_color_table(3G), inquire_color_table(3G), shade_mode(3G).

NAME

bank_switch – set graphics bank for multiple-byte-per-pixel frame buffers

SYNOPSIS

C Syntax:

```
void bank_switch(fildes,wbank,dbank);
int fildes,wbank,dbank;
```

FORTRAN77 Syntax:

```
subroutine bank_switch(fildes,wbank,dbank)
integer*4 fildes,wbank,dbank
```

Pascal Syntax:

```
procedure bank_switch(fildes,wbank,dbank:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

wbank Set graphics bank to be used for output primitives and raster operations.

dbank Set graphics bank to be displayed.

Discussion

The *bank_switch* parameter **wbank** defines:

- The bank used for byte-per-pixel output from *block_write*, *define_raster_echo*, and all output primitives,
- Data source bank for *block_read*,
- The bank where *block_move* is performed.

The **dbank** parameter defines which bank is to be displayed. Allowable value ranges for *wbank* and *dbank* are device-dependent, and may or may not be identical for both parameters. The lowest bank is numbered 0.

On a 24-plane graphics device, if in CMAP_FULL mode as set by *shade_mode*, if an output primitive color is specified with red, green and blue parameters, the output from the primitives is sent to all three banks.

Due to interactions between the two procedures, avoid using *bank_switch* when double buffering has been enabled by *double_buffer*.

On devices that support video blending the **dbank** parameter usage may be altered by blending control routines. See the *Starbase Device Drivers Library Manual* for details.

See the *Starbase Programming with X11* chapter on display control for more information about using this routine with a window system.

DEFAULTS

wbank = 0: enable lowest buffer for writing.
dbank = 0: enable lowest buffer for display.

SEE ALSO

double_buffer(3G), shade_mode(3G), *Starbase Device Drivers Library Manual, Starbase Graphics Techniques, Starbase Programming with X11.*

NAME

bf_control – activate or deactivate attributes for backfacing polygons.

SYNOPSIS

C Syntax:

```
void bf_control(fildes,rev_normals,attr);
int fildes,rev_normals,attr;
```

FORTRAN77 Syntax:

```
subroutine bf_control(fildes,rev_normals,attr)
integer*4 fildes,rev_normals,attr
```

Pascal Syntax:

```
procedure bf_control(fildes,rev_normals,attr:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

rev_normals If TRUE (1), any normals associated with polygons can point in either direction when light source equations are applied. The normal pointing more towards the light is used for computation.

attr If TRUE (1), backfacing polygons are rendered with the backface polygon attributes.

Discussion

The **rev_normals** flag is useful in the case where normals are being calculated on polygons whose vertices are not arranged in a consistently clockwise or counter-clockwise order (see *vertex_format* procedure). It is also useful for polygons that are to be viewed as two-sided surfaces.

If **attr** is TRUE, polygons that are backfacing with respect to the viewpoint are rendered by the light source equations using polygon or vertex normals pointing in the opposite direction and a new set of attributes. The attributes include the fill color specified by *bf_fill_color (3G)*, perimeter color specified by *bf_perimeter_color (3G)*, interior style specified by *bf_interior_style (3G)*, and the shading model specified by *bf_surface_model (3G)* and *bf_surface_coefficients (3G)*. If backface culling is enabled by the *hidden_surface*(3G) **cull** parameter, any subsequent call to *bf_control* with **attr** TRUE has no effect because backface culling remains active. See "Exceptions to Standard Starbase Support" in the *HP98721 Device Driver* section of the *Starbase Device Drivers Library Manual*.

This routine replaces *backface_control* which, although still supported, is not recommended because it modifies only the fill color of the back facing polygons.

DEFAULTS

rev_normals = FALSE (0).
attr = FALSE (0).

SEE ALSO

fill_color(3G), interior_style(3G), perimeter_color(3G), surface_model(3G), surface_coefficients(3G), hidden_surface(3G), *Starbase Device Drivers Library Manual.*

NAME

bitmap_print, dcbitmap_print, intbitmap_print – print bitmap contents on a raster printer

SYNOPSIS

C Syntax:

```
void bitmap_print(fildes,formatter,config,print_mode,full,xstart,
                  ystart,xlen,ylen,rotate,foreground,background,noback)
int fildes;
char *formatter,*config;
int print_mode,full;
float xstart,ystart;
int xlen,ylen;
int rotate,foreground,background,noback;

void intbitmap_print(fildes,formatter,config,print_mode,full,xstart,
                  ystart,xlen,ylen,rotate,foreground,background,noback)
int fildes;
char *formatter,*config;
int print_mode,full,xstart,ystart,xlen,ylen,rotate,foreground;
int background,noback;

void dcbitmap_print(fildes,formatter,config,print_mode,full,xstart,
                  ystart,xlen,ylen,rotate,foreground,background,noback)
int fildes;
char *formatter,*config;
int print_mode,full,xstart,ystart,xlen,ylen;
int rotate,foreground,background,noback;
```

FORTRAN77 Syntax:

```
subroutine bitmap_print(fildes,formatter,config,
                  print_mode,full,xstart,ystart,xlen,ylen,rotate,
                  foreground,background,noback)
integer*4 fildes
character*(*) formatter,config
integer*4 print_mode,full
real xstart,ystart
integer*4 xlen,ylen
integer*4 rotate,foreground,background,noback

subroutine intbitmap_print(fildes,formatter,config,
                  print_mode,full,xstart,ystart,xlen,ylen,rotate,
                  foreground,background,noback)
integer*4 fildes
character*(*) formatter,config
integer*4 print_mode,full,xstart,ystart,xlen,ylen
integer*4 rotate,foreground,background,noback

subroutine dcbitmap_print(fildes,formatter,config,
                  print_mode,full,xstart,ystart,xlen,ylen,
                  rotate,foreground,background,noback)
integer*4 fildes
character*(*) formatter,config
integer*4 print_mode,full,xstart,ystart,xlen,ylen
integer*4 rotate,foreground,background,noback
```

Pascal Syntax:

```
procedure bitmap_print(fildes:integer;formatter,config:string255;print_mode,
```

full:integer;xstart,ystart:real; xlen,ylen,rotate,
foreground,background,noback:integer);

procedure intbitmap_print(fildes:integer;formatter,config:string255;
print_mode,full,xstart,ystart,xlen,ylen,rotate,foreground,
background,noback:integer);

procedure dcbitmap_print(fildes:integer;formatter,config:string255;
print_mode,full,xstart,ystart,xlen,ylen,rotate,foreground,
background,noback:integer);

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when the I/O path of a bitmap device is opened.

formatter Character representation of the formatter to be selected. The only formatter supported by Starbase is the pcl ("pcl"). See *Starbase Device Drivers Library* (formatter sections) for details. If non-supported formatters are used, see the documentation supplied with them.

config Pathname of a configuration file where information pertaining to the raster formatter and output device will be found. The contents of this file are formatter-dependent.

print_mode Integer value indicating how the data is to be read. If **print_mode** is non-negative, it specifies a single plane to be printed; if it exceeds the number of planes available, a failure indication is returned. If **print_mode** is negative or equals ALL_PLANES (-1), all display-enabled planes are printed.

full If TRUE (1), indicates that the full extent of the bitmap is to be printed. The parameters **xstart, ystart, xlen,** and **ylen** are ignored. If FALSE (0), the rectangle specified by the next four parameters is printed.

xstart, ystart Specifies the upper-left corner of the source rectangle in virtual device coordinates (*bitmap_print, intbitmap_print*) or device coordinates (*dcbitmap_print*).

xlen, ylen Lengths (in pixels) of the sides of the rectangle to be printed.

rotate If TRUE (1), indicates that the formatter should rotate the rectangle 90 degrees for output (the direction of rotation is formatter-dependent). If FALSE (0), the formatter-dependent default orientation is used.

foreground, background
Specify color map indices to be used when printing a single plane. The color corresponding to **foreground** is printed for **one**s in the source, and the color corresponding to **background** is printed for **zero**es (however, see the **noback** parameter, below).

noback If TRUE (1), the color map index specified by **background** is mapped to no color; that is, it is not printed. This parameter applies, whether a single plane or all planes are being printed. If FALSE (0), the background color in the bitmap (or the specified background color if a single plane) is printed just as any other color.

Discussion

Pixel data is considered to be an **xlen** by **ylen** pixel rectangle for the following discussion. The pixel data from the source is read, beginning at upper-left corner **xstart,ystart** for the entire rectangle defined by **xlen,ylen**; however, if the **full** parameter is TRUE, the rectangle is determined by the bitmap extent. The source is clipped against the current clip rectangle.

RGB information is converted by the formatter (through dithering) according to the capabilities of the output device. Actual data written is controlled by the formatter and the configuration file. The formatter may clip the data as necessary to fit the output medium.

Only currently displayed planes are printed. Non-displayed planes are interpreted as zeroes. If double-buffering is enabled, the currently displayed buffer is printed.

For all-planes operation with multi-bank graphics devices, the currently selected bank (as set by *bank_switch*) is printed. If the current shade mode is CMAP_FULL, all appropriate banks are printed.

Conditions under which the operation fails include specification of an invalid formatter string (one not found in the formatter table linked with the program); and conditions under which the formatter itself returns a failure indication. *Inquire_gerror*(3G) can be used to determine the error.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

SEE ALSO

file_print(3G), *Starbase Device Drivers Library Manual.*

NAME

bitmap_to_file, dcbitmap_to_file, intbitmap_to_file – copy bitmap contents to a bitmap file

SYNOPSIS

C Syntax:

```
void bitmap_to_file(fildes,full_depth,spn,dpn,dest,full,
                    xstart,ystart,xlen,ylen,store_cmap)
int fildes,full_depth,spn,dpn;
char *dest;
int full;
float xstart,ystart;
int xlen,ylen;
int store_cmap;

void dcbitmap_to_file(fildes,full_depth,spn,dpn,dest,
                    full,xstart,ystart,xlen,ylen,store_cmap)
int fildes,full_depth,spn,dpn;
char *dest;
int full,xstart,ystart,xlen,ylen;
int store_cmap;

void intbitmap_to_file(fildes,full_depth,spn,dpn,dest,
                    full,xstart,ystart,xlen,ylen,store_cmap)
int fildes,full_depth,spn,dpn;
char *dest;
int full,xstart,ystart,xlen,ylen,store_cmap;
```

FORTRAN77 Syntax:

```
subroutine bitmap_to_file(fildes,full_depth,spn,dpn,dest,
                    full,xstart,ystart,xlen,ylen,store_cmap)
integer*4 fildes,full_depth,spn,dpn
character*(*) dest
integer*4 full
real xstart,ystart
integer*4 xlen,ylen
integer*4 store_cmap

subroutine dcbitmap_to_file(fildes,full_depth,spn,dpn,dest,
                    full,xstart,ystart,xlen,ylen,store_cmap)
integer*4 fildes,full_depth,spn,dpn
character*(*) dest
integer*4 full,xstart,ystart,xlen,ylen
integer*4 store_cmap

subroutine intbitmap_to_file(fildes,full_depth,spn,dpn,dest,
                    full,xstart,ystart,xlen,ylen,store_cmap)
integer*4 fildes,full_depth,spn,dpn
character*(*) dest
integer*4 full,xstart,ystart,xlen,ylen,store_cmap
```

Pascal Syntax:

```
procedure bitmap_to_file(fildes,full_depth,spn,dpn:integer;
                    dest:string255;full:integer;xstart,ystart:real;
                    xlen,ylen,store_cmap:integer);

procedure dcbitmap_to_file(fildes,full_depth,spn,dpn:integer;
                    dest:string255;full,xstart,ystart,xlen,ylen,
```

store_cmap:integer);

procedure intbitmap_to_file(fildes,full_depth,spn,dpn:integer;
dest:string255;full,xstart,ystart,xlen,ylen,
store_cmap:integer);

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when the I/O path of a bitmap device is opened.

full_depth Indicates what source data is to be stored. If TRUE (1), all bitmap planes are transferred to the file, ignoring **spn** and **dpn** parameters. If FALSE (0), the parameter **spn** determines a single plane to be transferred.

spn Number of a single plane to copy to the file. It must be in the range 0 to $n-1$, where n is the number of planes in the bitmap.

dpn Destination plane number recorded in the file; can be any non-negative integer. Any plane number can be assigned to the data stored in the file. To mark the destination plane number to be the same as the source plane, set **spn** and **dpn** to the same value.

dest Pathname of the file to be written. If a file by the same name exists, it is overwritten by the new file.

full If TRUE (1), indicates that the full extent of the bitmap is to be stored. The parameters **xstart, ystart, xlen,** and **ylen** are ignored. If FALSE (0), the rectangle specified by the next four parameters is stored.

xstart, ystart Specify coordinates of the upper-left corner of the source rectangle in virtual device coordinates (*bitmap_to_file*, *intbitmap_to_file*) or device coordinates (*dcbitmap_to_file*).

xlen, ylen Lengths, in pixels, of the sides of the rectangle to be stored.

store_cmap If TRUE (1), indicates that the current software color map is to be stored in the file. If FALSE (0), no color map is stored.

Discussion

Pixel data is considered to be an **xlen** by **ylen** pixel rectangle for the following discussion. The pixel data from the source beginning at upper left corner **xstart,ystart** is written into the destination file. If the **full** parameter is TRUE, the rectangle is determined by the bitmap extent.

If the file is written full-depth from a bitmap organized in pixel-major format, the file will be written in a pixel-major format. If the copy is full-depth from a plane-major bitmap, the file will be plane_major also. A single-plane copy always creates a plane_major file.

The actual rectangle written to the file is determined by the current clipping mode and hence a later call to *inquire_file* may return smaller dimensions than were specified in the procedure call that created the file.

Only currently displayed planes are stored. Non-displayed planes are stored as zeroes. If double-buffering is enabled, the currently displayed buffer is stored.

For full-depth operation from multi-bank graphics devices, the currently selected bank (as set by *bank_switch* is stored. If the current shade mode is CMAP_FULL, all appropriate banks are stored.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer

operations, floating point operations and common operations see the *starbase.3g* manual page.

SEE ALSO

bitmap_print(3G), file_to_bitmap(3G), file_print(3G), inquire_file(3G).

NAME

block_move, dcblock_move, intblock_move – frame buffer to frame buffer copy procedure.

SYNOPSIS

C Syntax:

```
void block_move(fildes,x_source,y_source,length_x,length_y,x_dest,y_dest);
int fildes;
float x_source,y_source,x_dest,y_dest;
int length_x,length_y;

void dcblock_move(fildes,x_source,y_source,length_x,length_y,x_dest,
        y_dest);
int fildes;
int x_source,y_source,length_x,length_y,x_dest,y_dest;

void intblock_move(fildes,x_source,y_source,length_x,length_y,x_dest,
        y_dest);
int fildes,x_source,y_source,length_x,length_y,x_dest,y_dest;
```

FORTRAN77 Syntax:

```
subroutine block_move(fildes,x_source,y_source,length_x,length_y,
        x_dest,y_dest)
integer*4 fildes
real x_source,y_source,x_dest,y_dest
integer*4 length_x,length_y

subroutine dcblock_move(fildes,x_source,y_source,length_x,length_y,
        x_dest,y_dest)
integer*4 fildes,x_source,y_source,x_dest,y_dest,length_x,length_y

subroutine intblock_move(fildes,x_source,y_source,length_x,length_y,
        x_dest,y_dest)
integer*4 fildes,x_source,y_source,length_x,length_y,x_dest,y_dest
```

Pascal Syntax:

```
procedure block_move(fildes:integer;x_source,y_source:real;
        length_x,length_y:integer;x_dest,y_dest:real);

procedure dcblock_move(fildes:integer;x_source,y_source,
        length_x,length_y,x_dest,y_dest:integer);

procedure intblock_move(fildes:integer;x_source,y_source,
        length_x,length_y,x_dest,y_dest:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

x_source,y_source Coordinates of upper left corner of the source in virtual device coordinates (*block_move intblock_move*) or device coordinates *(dcblock_move)*.

x_dest,y_dest Coordinates of upper left corner of destination in virtual device coordinates or device coordinates.

length_x,length_y Length (in pixels) of the horizontal and vertical sides of the block being moved.

Discussion

Block_move copies the source rectangle beginning at location (x_source,y_source) with sides of length_x and length_y to the destination rectangle beginning at location (x_dest,y_dest) with sides of length_x and length_y using the current drawing mode.

If the clip indicator is not CLIP_OFF, or if the rectangle extends outside the raster limits of the display device, the block copy is limited to the intersection of the source rectangle and the current clip boundaries combined with the intersection of the destination window and the raster limits.

Only write enabled planes are affected.

On multi-bank graphics devices, block copy occurs only in the current bank set by *bank_switch*. If *shade_mode* is CMAP_FULL, the block move is performed on all displayed graphics banks, simultaneously.

Block_move does not copy between graphics devices.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

SEE ALSO

bank_switch(3G), block_read(3G), block_write(3G), clip_indicator(3G), clip_rectangle(3G), drawing_mode(3G), write_enable(3G).

NAME

block_read, dcblock_read, intblock_read – frame buffer to main memory block transfer procedure.

SYNOPSIS

C Syntax:

```
void block_read(fildes,x_source,y_source,length_x,length_y,
        pixel_data,raw);
int fildes;
float x_source,y_source;
int length_x,length_y,raw;
unsigned char pixel_data[];

void dcblock_read(fildes,x_source,y_source,length_x,length_y,
        pixel_data,raw);
int fildes,x_source,y_source,length_x,length_y,raw;
unsigned char pixel_data[];

void intblock_read(fildes,x_source,y_source,length_x,length_y,
        pixel_data,raw);
int fildes,x_source,y_source,length_x,length_y,raw;
unsigned char pixel_data[];
```

FORTRAN77 Syntax:

```
subroutine block_read(fildes,x_source,y_source,length_x,length_y,
        pixel_data,raw)
integer*4 fildes
real x_source,y_source
integer*4 length_x,length_y,raw
character*(*) pixel_data()

subroutine dcblock_read(fildes,x_source,y_source,length_x,length_y,
        pixel_data,raw)
integer*4 fildes,x_source,y_source,length_x,length_y,raw
character*(*) pixel_data()

subroutine intblock_read(fildes,x_source,y_source,length_x,length_y,
        pixel_data,raw)
integer*4 fildes,x_source,y_source,length_x,length_y,raw
character*(*) pixel_data()
```

Pascal Syntax:

```
type
        gbyte=0..255;

procedure block_read(fildes:integer;
        x_source,y_source:real;length_x,length_y:integer;
        var pixel_data:packed array[lo..hi:integer] of gbyte;raw:integer);

procedure dcblock_read(fildes:integer;
        x_source,y_source,length_x,length_y:integer;
        var pixel_data:packed array[lo..hi:integer] of gbyte;raw:integer);

procedure intblock_read(fildes:integer;
        x_source,y_source,length_x,length_y:integer;
        var pixel_data:packed array[lo..hi:integer] of gbyte;raw:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

x_source,y_source
Coordinates of upper left corner of the source in virtual device coordinates (*block_read, intblock_read*) or device coordinates (*dcblock_read*).

length_x,length_y
Length (in pixels) of the x and y sides of the block to be read.

raw If FALSE (0), each pixel is written as a 1 byte quantity. There are no padding bytes. Therefore, with FORTRAN, four pixels are written for each 32-bit integer array element.

If TRUE (1), the data is organized in a device dependent format. Some devices may format the device with multiple pixels per byte. Other devices may allow different formats that can be set with a call to the *gescape* **R_BIT_MODE**. Other functions, such as clipping, may also behave in a device dependent manner. See the *Starbase Device Driver Library* for details.

Output Parameters

pixel_data Packed array of pixel data to be transferred. This data is returned by the procedure.

Discussion

Pixel_data is considered to be a length_x by length_y byte rectangle for the following discussion. The returned array will contain frame buffer values from the source rectangle beginning at location (x_source,y_source) with dimensions of length_x and length_y combined with the values in pixel_data using the current *drawing_mode*. *Write_enable* does not have any effect on *block_read*. Pixels are placed in the array in order: left to right, then from top to bottom. If the *clip_indicator* is not CLIP_OFF, *pixel_data* values outside the intersection of the source rectangle and the current clip boundaries are left untouched. In all cases the dimensions of the resultant *pixel_data* rectangle is preserved.

This is the exact inverse operation of *block_write* (except for *write_enable*).

For multi-bank graphics devices the block read comes from the bank currently selected by *bank_switch*.

For device-independent operation do not use raw mode. Raw mode should only be used where speed of image read is critical and the number of pixels read is not critical. This speed improvement is significant only if the device frame buffer organization is not one byte or word per pixel.

Block_read and *intblock_read* are performed using virtual device coordinate values and *dcblock_read* is performed using device coordinate values.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

SEE ALSO

bank_switch(3G), block_move(3G), block_write(3G), clip_indicator(3G), clip_rectangle(3G), drawing_mode(3G), write_enable(3G).

NAME

block_write, dcblock_write, intblock_write – main memory to frame buffer block transfer procedure.

SYNOPSIS

C Syntax:

```
void block_write(fildes,x_dest,y_dest,length_x,length_y,pixel_data,raw);
        int fildes;
        float x_dest,y_dest;
        int length_x,length_y,raw;
        unsigned char pixel_data[];
void intblock_write(fildes,x_dest,y_dest,length_x,length_y,pixel_data,raw);
        int fildes,x_dest,y_dest,length_x,length_y,raw;
        unsigned char pixel_data[];
void dcblock_write(fildes,x_dest,y_dest,length_x,length_y,pixel_data,raw);
        int fildes,x_dest,y_dest,length_x,length_y,raw;
        unsigned char pixel_data[];
```

FORTRAN77 Syntax:

```
subroutine block_write(fildes,x_dest,y_dest,length_x,length_y,pixel_data,raw)
integer*4 fildes
real x_dest,y_dest
integer*4 length_x,length_y,raw
character*(*) pixel_data()

subroutine intblock_write(fildes,x_dest,y_dest,length_x,length_y,pixel_data,raw)
integer*4 fildes,x_dest,y_dest,length_x,length_y,raw
character*(*) pixel_data()

subroutine dcblock_write(fildes,x_dest,y_dest,length_x,length_y,pixel_data,raw)
integer*4 fildes,x_dest,y_dest,length_x,length_y,raw
character*(*) pixel_data()
```

Pascal Syntax:

```
type
gbyte=0..255;

procedure block_write(fildes:integer;x_dest,y_dest:real;length_x,length_y:integer;
        var pixel_data:packed array[lo..hi:integer] of gbyte;raw:integer);

procedure intblock_write(fildes:integer;x_dest,y_dest,length_x,length_y:integer;
        var pixel_data:packed array[lo..hi:integer] of gbyte;raw:integer);

procedure dcblock_write(fildes:integer;x_dest,y_dest,length_x,length_y:integer;
        var pixel_data:packed array[lo..hi:integer] of gbyte;raw:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

x_dest,y_dest Coordinates of upper left corner of the destination in virtual device coordinates (*block_write, intblock_write*), or device coordinates (*dcblock_write*).

length_x,length_y
Length of the sides of the block to be written in pixels.

pixel_data Packed array of pixel data to be transferred (written) to the frame buffer by the procedure.

raw If FALSE (0), each pixel is written as a 1-byte quantity. There are no padding bytes. Therefore, with FORTRAN, four pixels are written for each 32 bit integer array element.

If TRUE (1), data is organized in a device-dependent format. Some devices may format the device with multiple pixels per byte. Other devices may allow different formats that can be set with a call to the *gescape* R_BIT_MODE. Some devices can get speed improvement by using the *gescape* R_DMA_MODE. Other functions, such as clipping, may also behave in a device dependent manner. See the *Starbase Device Driver Library Manual* for details.

Discussion

Pixel data is considered to be a **length_x** by **length_y** byte rectangle for the following discussion. The **pixel_data** array contains values that are to be combined with the frame buffer destination rectangle beginning at location **(x_dest,y_dest)** having dimensions of **length_x** and **length_y** using the current *drawing_mode* and *write_enable*. Pixels are read from the array in order: left to right, then top to bottom. If the *clip_indicator* is not CLIP_OFF, frame buffer values outside the intersection of the destination rectangle and the current clip boundaries are left untouched. In all cases the dimensions of the pixel_data source rectangle is preserved.

This is the exact inverse operation of *block_read* (except for *write_enable*).

For multi-plane graphics devices, the block write goes to the currently selected bank as set by *bank_switch*.

For device independent operation do not use raw mode. Raw mode should only be used where speed of image written is critical and the number of pixels written is not critical. This speed improvement is significant only if the device frame buffer organization is not a byte or word per pixel, or if using the *gescape* R_DMA_MODE.

Block_write and *intblock_write* are done using Virtual Device Coordinate values and *dcblock_write* is done using Device Coordinate values.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that **fildes**. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase(3G)* manual page.

SEE ALSO

bank_switch(3G), block_move(3G), block_read(3G), clip_indicator(3G), clip_rectangle(3G), drawing_mode(3G), write_enable(3G).

NAME

buffer_mode – set buffering mode for output primitives.

SYNOPSIS

C Syntax:

```
void buffer_mode(fildes,on);
int fildes,on;
```

FORTRAN77 Syntax:

```
subroutine buffer_mode(fildes,on)
integer*4 fildes,on
```

Pascal Syntax:

```
procedure buffer_mode(fildes,on;integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

on If **on** = TRUE, buffering mode is enabled;
If **on** = FALSE, buffering mode is disabled;

Discussion

Buffer_mode is used to enable or disable buffering of output data for graphic output primitives. Typically, device drivers accumulate output primitives in an output buffer, then, when the buffer is full or a *make_picture_current* is performed, the device is locked to prevent other processes from interfering, the buffer is processed, then the device is unlocked. With buffering disabled, *make_picture_current* is called automatically after every Starbase output primitive procedure. The overhead involved in locking and unlocking the device can substantially impair system performance when buffering is disabled. A more efficient way to ensure an updated display is to invoke *make_picture_current* after a group of primitives have been processed. Disabling buffering is sometimes helpful when debugging a program that uses Starbase routines.

DEFAULTS

Buffering: **on** (TRUE).

SEE ALSO

Make_picture_current(3G)

NAME

cgm_to_starbase – interpret a cgm picture

SYNOPSIS

C Syntax:

```
void cgm_to_starbase(fildes,source,picture_number);
int fildes,picture_number;
char *source;
```

FORTRAN77 Syntax:

```
subroutine cgm_to_starbase(fildes,source,picture_number)
integer*4 fildes,picture_number
character*(*) source
```

Pascal Syntax:

```
procedure cgm_to_starbase (fildes: integer;var source: string255;
        picture_number: integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

source Pathname of the cgm file.

picture_number
Picture number in metafile to be interpreted.

Discussion

CGM is a graphical data base suitable for the storage and retrieval of picture information in a way that is compatible between devices of differing capabilities and design.

Cgm_to_starbase reads the cgm file specified by **source** and interprets pictures in the metafile by converting cgm elements into starbase. The picture, **picture_number**, is interpreted and output to the device specified by **fildes**. The first picture in the cgm corresponds to **picture_number**, equal to one. The second picture in the cgm corresponds to **picture_number**, equal to two, and so on up to the number of pictures in the metafile. *inquire_cgm* can be called to obtain the number of pictures in the metafile as well as other information about the metafile.

/usr/lib/libsbcgm.a must be linked to this program. See CGM in the *Starbase Graphics Techniques* manual

SEE ALSO

inquire_cgm(3G), *Starbase Device Drivers Library Manual, Starbase Graphics Techniques.*

NAME

character_expansion_factor – set character cell height-to-width ratio.

SYNOPSIS

C Syntax:

```
void character_expansion_factor(fildes,factor);
int fildes;
float factor;
```

FORTRAN77 Syntax:

```
subroutine character_expansion_factor(fildes,factor)
integer*4 fildes
real factor
```

Pascal Syntax:

```
procedure character_expansion_factor(fildes:integer;factor:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

factor Deviation factor from the width/height ratio specified with each font.

Discussion

Character_expansion_factor (relative character width) sets the character cell width to
(factor)*(character height)*(font width/height ratio)

The *character_width* and *character_expansion_factor* procedures both modify the width of a character. The procedure most recently executed defines the character width for subsequently drawn characters.

DEFAULTS

Factor = 1.0.

SEE ALSO

character_width(3G), character_height(3G), intra_character_space(3G), *Starbase Graphics Techniques*.

NAME

character_height, dccharacter_height, intcharacter_height – set character height.

SYNOPSIS

C Syntax:

```
void character_height(fildes,height);
int fildes;
float height;

void dccharacter_height(fildes,dcheight);
int fildes,dcheight;

void intcharacter_height(fildes,height);
int fildes,height;
```

FORTRAN77 Syntax:

```
subroutine character_height(fildes,height)
integer*4 fildes
real height

subroutine dccharacter_height(fildes,dcheight)
integer*4 fildes,dcheight

subroutine intcharacter_height(fildes,height)
integer*4 fildes,height
```

Pascal Syntax:

```
procedure character_height(fildes:integer;height:real);

procedure dccharacter_height(fildes,dcheight:integer);

procedure intcharacter_height(fildes,height:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

height Character height from bottom to top along the character *up vector* in virtual device coordinates. The up vector orients the character.

dcheight Integer height of the character from bottom to top along the character **up** vector in device coordinates.

Discussion

Character_height and *intcharacter_height* specify the height of subsequently drawn characters. The height is measured along the character **up** vector from the character cell bottom to the character cell top.

Dccharacter_height specifies the height of subsequently drawn device coordinate characters.

Note: If *vdc_extent* or *intvdc_extent* are changed, the character height changes with the new VDC extent. The default VDC character height is used if no previous calls were made to *character_height* or *intcharacter_height*.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

DEFAULTS

1/50 of the y-dimension of the default virtual device coordinate extent. The size of the

characters will change if the character **up** vector is not vertical with respect to the output device's display.

For dccharacter_height, 15 device coordinates.

SEE ALSO

character_width(3G), character_expansion_factor(3G), vdc_extent(3G), *Starbase Graphics Techniques*.

NAME

character_slant – specify character slant

SYNOPSIS

C Syntax:

```
void character_slant(fildes,slant);
int fildes;
float slant;
```

FORTRAN77 Syntax:

```
subroutine character_slant(fildes,slant)
integer*4 fildes
real slant
```

Pascal Syntax:

```
procedure character_slant(fildes:integer;slant:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

slant Tangent of the angle between the character **up** vector and the trailing edge of the character box measured in a clockwise direction.

Discussion

Character_slant specifies the character slant for all subsequently drawn characters. The location of the character's origin, the baseline-to-capline height, and the length of the character baseline are not affected by *character_slant.*

A slanted point can be calculated using the equation:

$$x = x + y * slant$$

where (x,y) is any point in the character body (y=0 is the baseline). Slant affects only the x value; the y value is unchanged.

DEFAULTS

Zero slant

SEE ALSO

text_orientation(3G)

NAME

character_width, dccharacter_width, intcharacter_width – specify character width

SYNOPSIS

C Syntax:

```
void character_width(fildes,width);
int fildes;
float width;

void dccharacter_width(fildes,dcwidth);
int fildes,dcwidth;

void intcharacter_width(fildes,width);
int fildes,width;
```

FORTRAN77 Syntax:

```
subroutine character_width(fildes,width)
integer*4 fildes
real width

subroutine dccharacter_width(fildes,dcwidth)
integer*4 fildes,dcwidth

subroutine intcharacter_width(fildes,width)
integer*4 fildes,width
```

Pascal Syntax:

```
procedure character_width(fildes:integer;width:real);

procedure dccharacter_width(fildes,dcwidth:integer);

procedure intcharacter_width(fildes,width:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

width Width of a text character measured in virtual device coordinates along the baseline.

dcwidth Width of a text character measured in device coordinates along the baseline.

Discussion

Character_width or *intcharacter_width* specify the width of subsequently drawn characters. The *character_expansion_factor*, *character_width*, and *intcharacter_width* procedures modify the width of a character. The procedure most recently executed defines the character width for subsequently drawn characters.

Character width values set by *character_width* or *intcharacter_width* are independent of character height values.

Beware:

If *vdc_extent* is changed, the character width is recomputed using the new VDC extent and the last character width set, or the default if no previous calls to *character_width* were made.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

DEFAULTS

None, character width is defined by the default character expansion factor of 1.0.

SEE ALSO

character_expansion_factor(3G), character_height(3G), text(3G), vdc_extent(3G).

NAME

intcircle, intpartial_circle, dccircle – define a circular region to be filled and/or edged. (See *ellipse.3g* for floating point circle information).

SYNOPSIS

C Syntax:

```
void intcircle (fildes,radius,x_center,y_center);
int fildes,radius,x_center,y_center;

void intpartial_circle (fildes,radius,x_center,y_center,closure);
int fildes,radius,x_center,y_center,closure;

void dccircle (fildes,dcradius,dcx_center,dcy_center);
int fildes,dcradius,dcx_center,dcy_center;
```

FORTRAN77 Syntax:

```
subroutine intcircle (fildes,radius,x_center,y_center)
integer*4 fildes,radius,x_center,y_center

subroutine intpartial_circle (fildes,radius,x_center,y_center,closure)
integer*4 fildes,radius,x_center,y_center,closure

subroutine dccircle (fildes,dcradius,dcx_center,dcy_center)
integer*4 fildes,dcradius,dcx_center,dcy_center
```

Pascal Syntax:

```
procedure intcircle (fildes,radius,x_center,y_center:integer);

procedure intpartial_circle (fildes,radius,x_center,y_center,
        closure:integer);

procedure dccircle (fildes,dcradius,dcx_center,dcy_center:integer);
```

DESCRIPTION

Input Parameters

fildes	Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.
radius,dcradius	Radius of the circle.
x_center,y_center	Coordinate of the circle center.
dcx_center,dcy_center	Device coordinate of the circle center.
closure	If TRUE (1), and the polygon vertex list is non-empty and the last subpolygon was not yet closed, a boundary is added to close that subpolygon with a boundary. If FALSE (0), no boundary is added.

Discussion

A circle of the specified radius, centered at (**x_center, y_center**) or (**dcx_center, dcy_center**) is filled and/or outlined according to the current interior style. As with all output primitives, the arc is affected by the current drawing mode and write enable. The curve is drawn with chords of length specified in *curve_resolution*. For *intpartial_circle*, the vertices of the circle are added to the polygon vertex list, as in the case of *partial_polygons*.

Non-filled circles are generated by setting the interior style to INT_HOLLOW and edged.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode Floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of Integer operations, Floating point operations and common operations see the *starbase.3g* manual page.

NOTE

Doing any other Starbase call besides *intpartial_circle* or *intpartial_polygon* in the middle of a list of procedure calls that add vertices to the polygon buffer and before a call to *intpolygon*, *intrectangle*, *intarc*, or *intcircle* produces unpredictable, device-dependent results.

When using a distorted mapping (x units do not equal y units), integer circles may not remain circular. This effect is unpredictable and device-dependent, and may not remain consistent with future releases.

In some cases, circles may actually be rendered as polygons. Certain devices have a limit on the number of polygon vertices they support (see the *Starbase Device Driver's Manual* section for your particular device). If this limit is exceeded, you will receive a warning. To eliminate this warning, adjust the step size using the *curve_resolution* command.

SEE ALSO

curve_resolution(3G), drawing_mode(3G), fill_color(3G), write_enable(3G), *Starbase Graphics Techniques*.

NAME

clear_control – select type of clearing for subsequent *clear_view_surface* procedures

SYNOPSIS

C Syntax:

```
void clear_control(fildes,mode);
int fildes,mode;
```

FORTRAN77 Syntax:

```
subroutine clear_control(fildes,mode)
integer*4 fildes,mode
```

Pascal Syntax:

```
procedure clear_control(fildes,mode:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

mode The area to be cleared is specified with one of the following values:

CLEAR_VDC_EXTENT	Clear virtual device coordinate extent area.
CLEAR_CLIP_RECTANGLE	Clear clip rectangle area.
CLEAR_DISPLAY_SURFACE	Clear entire display surface.
CLEAR_VIEWPORT	Clear current viewport area.

In addition, the above values may be ORed with one or more of the following control flags:

CLEAR_ALL_BANKS	For multibank devices, clear all graphics banks to zero simultaneously.
CLEAR_ZBUFFER	Clear the zbuffer (if hidden surface removal is enabled) simultaneously with screen clears.

Discussion

For continuous-feed hardcopy devices, CLEAR_DISPLAY_SURFACE ejects the current page.

For multibank graphics devices, only the current bank as set by *bank_switch* is cleared. If CLEAR_ALL_BANKS is ORed with *mode*, all graphics banks are cleared to zero simultaneously, independent of *write_enable*, *bank_switch*, and *background_color*.

For devices supporting zbuffer hidden surface removal, the zbuffer will be cleared when *clear_view_surface* is called if CLEAR_ZBUFFER is ORed with *mode*. On devices which support zbuffer clearing at the same time as screen clearing, performance will be increased if the zbuffer is cleared in this manner, instead of using a separate call to *zbuffer_switch*. When CLEAR_ZBUFFER is enabled, *zbuffer_switch* will NOT clear the zbuffer.

DEFAULTS

Clear entire display surface. Do not clear all graphics banks; do not clear the zbuffer.

SEE ALSO

background_color(3G), bank_switch(3G), clear_view_surface(3G), clip_rectangle(3G), set_p1_p2(3G), vdc_extent(3G), view_port(3G), zbuffer_switch(3G), *Starbase Device Drivers Library Manual.*

NAME

clear_view_surface – set all or part of physical view surface to *background_color*

SYNOPSIS

C Syntax:

```
void clear_view_surface(fildes);
int fildes;
```

FORTRAN77 Syntax:

```
subroutine clear_view_surface(fildes)
integer*4 fildes
```

Pascal Syntax:

```
procedure clear_view_surface(fildes:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

Discussion

Clear_view_surface repaints a portion of the view surface as indicated by *clear_control* with the index set by the last *background_color* call. Only planes that are write enabled are affected. A replacement rule of **3** (source) is used, regardless of the rule set with *drawing_mode*.

For multi-bank graphics devices in the default mode, only the current bank as selected by *bank_switch* is cleared. All banks can be cleared simultaneously with the proper call to *clear_control*.

Continuous-feed hardcopy devices may eject a new page, depending on the setting of *clear_control*.

SEE ALSO

background_color(3G), bank_switch(3G), clear_control(3G), drawing_mode(3G), write_enable(3G).

NAME

clip_depth – define front and back clipping planes

SYNOPSIS

C Syntax:

```
void clip_depth(fildes,front,back);
int fildes;
float front,back;
```

FORTRAN77 Syntax:

```
subroutine clip_depth(fildes,front,back)
integer*4 fildes
real front,back
```

Pascal Syntax:

```
procedure clip_depth(fildes:integer;front,back:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when the I/O path to a graphic device is opened.

front specifies the front clipping plane in perspective vdc coordinates; i.e., clipping is performed after the primitives are transformed by the current transformation matrix from modelling coordinates to vdc coordinates. Perspective divide occurs after the front and back clip.

back specifies the back clipping plane in perspective vdc coordinates; i.e., clipping is performed after the primitives are transformed by the current transformation matrix from modelling coordinates to vdc coordinates. Perspective divide is performed after the front and back clip.

Discussion

Clip_depth specifies the front and back clipping planes. Front must be less than back. *Clip_depth* does not enable/disable front or back clipping. Clipping to these planes is enabled or disabled using the **front_on,back_on** parameter of the *depth_indicator* procedure. This procedure also determines the range of transformed z values. This range is useful when implementing hidden surface removal procedures.

Note: *Clip_depth* values are also reset to the full z extent of the virtual device coordinates whenever *vdc_extent* is changed. This is a change present in all revisions after, and including the 5.18 release(on series 300) and is different from previous behavior!!!

DEFAULTS

front = 0.0, **back** = 1.0.

SEE ALSO

depth_indicator(3G), clip_indicator(3G), clip_rectangle(3G), vdc_extent(3G).

NAME

clip_indicator – enable/disable clipping to clip rectangle and virtual device coordinate extent

SYNOPSIS

C Syntax:

```
void clip_indicator (fildes,clip_level)
int fildes,clip_level;
```

FORTRAN77 Syntax:

```
subroutine clip_indicator(fildes,clip_level)
integer*4 fildes,clip_level
```

Pascal Syntax:

```
procedure clip_indicator(fildes,clip_level:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when the I/O path to the output graphic device is opened.

clip_level Determines whether clipping to clip_rectangle (CLIP_TO_RECT), virtual device coordinate extent (CLIP_TO_VDC), Viewport (CLIP_TO_VIEWPORT), or not at all (CLIP_OFF).

Discussion

The *clip_level* parameter can be one of the following:

CLIP_TO_RECT	Clipping is performed to the intersection of clip rectangle and virtual device coordinate extent.
CLIP_TO_VDC	Clipping is performed to the virtual device coordinate extent boundary.
CLIP_TO_VIEWPORT	Clipping is performed to the current viewport area.
CLIP_OFF	No clipping is performed; i.e., output primitives are not clipped at all (this may result in device-dependent wraparound or errors if primitives need to be clipped).

CLIP_OFF should only be used when the output primitives are known to be within the hard clip area. Performance will be substantially better if *clip_indicator* is set to CLIP_OFF.

DEFAULTS

clip_level = CLIP_TO_RECT

SEE ALSO

clip_depth(3G), clip_rectangle(3G), depth_indicator(3G), set_p1_p2(3G), vdc_extent(3G), view_port(3G).

NAME

clip_rectangle, intclip_rectangle – define current clip rectangle boundaries

SYNOPSIS

C Syntax:

```
void clip_rectangle(fildes,lower_left_x,upper_right_x,lower_left_y,
        upper_right_y);
int fildes;
float lower_left_x,upper_right_x,lower_left_y,upper_right_y;

void intclip_rectangle(fildes,lower_left_x,upper_right_x,lower_left_y,
        upper_right_y);
int fildes,lower_left_x,upper_right_x,lower_left_y,upper_right_y;
```

FORTRAN77 Syntax:

```
subroutine clip_rectangle(fildes,lower_left_x,upper_right_x,lower_left_y,
        upper_right_y)
integer*4 fildes
real lower_left_x,upper_right_x,lower_left_y,upper_right_y

subroutine intclip_rectangle(fildes,lower_left_x,upper_right_x,
        lower_left_y,upper_right_y)
integer*4 fildes,lower_left_x,upper_right_x,lower_left_y,upper_right_y
```

Pascal Syntax:

```
procedure clip_rectangle(fildes:integer;lower_left_x,upper_right_x,
        lower_left_y,upper_right_y:real);

procedure intclip_rectangle(fildes,lower_left_x,upper_right_x,lower_left_y,
        upper_right_y:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when the I/O path to the output device is opened.

lower_left_x,lower_left_y,upper_right_x,upper_right_y
Coordinate values defining the lower left and upper right corners of a rectangular extent in virtual device coordinate space, thus defining the clip rectangle.

Discussion

When **clip_level** is set to CLIP_TO_RECT by using *clip_indicator*, output primitives are clipped to the intersection of this rectangular region and the virtual device coordinate extent.

The results of this procedure are also used by *clear_view_surface* when clear control is set to CLEAR_CLIP_RECTANGLE.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

Note: The clip rectangle area is reset to be the entire *vdc_extent* area whenever *vdc_extent* or *intvdc_extent* is changed. This is a change present in all revisions beginning with and including HP-UX Release 5.18 (on Series 300 Model 320), and is different from previous behavior.

DEFAULTS

Float: **(lower_left_x,upper_right_x,lower_left_y,upper_right_y)=(0.0,1.0,0.0,1.0)**
Integer: **(lower_left_x,upper_right_x,lower_left_y,upper_right_y)=(0,32767,0,32767)**

SEE ALSO

clear_view_surface(3G), clip_indicator(3G), clip_depth(3G), depth_indicator(3G), vdc_extent(3G).

NAME

concat_matrix, intconcat_matrix2d – multiply two matrices and return the resulting matrix

SYNOPSIS

C Syntax:

```
void concat_matrix(matrix1,matrix2,result);
float matrix1[4][4],matrix2[4][4],result[4][4];

void intconcat_matrix2d(matrix1,matrix2,result,radix1,radix2,radix,raw);
int matrix1[3][2],matrix2[3][2],result[3][2],radix1,radix2,*radix,raw;
```

FORTRAN77 Syntax:

(See Language Dependencies Below)

```
subroutine concat_matrix(matrix1,matrix2,result)
real matrix1(4,4),matrix2(4,4),result(4,4)

subroutine intconcat_matrix2d(matrix1,matrix2,result,
                radix1,radix2,radix,raw)
integer*4 matrix1(2,3),matrix2(2,3),result(2,3),radix1,radix2,radix,raw
```

Pascal Syntax:

```
type
    three_d_xform = array[1..4][1..4] of real;
    int2d_xform = array[1..3][1..2] of integer;

procedure concat_matrix(var matrix1,matrix2,result:three_d_xform);

procedure intconcat_matrix2d(var matrix1,matrix2,result:int2d_xform;
    radix1,radix2:integer;var radix:integer; raw:integer);
```

DESCRIPTION

Input Parameters

matrix1,matrix2
: Two user-supplied 4x4 real matrices presented in row major form, or two user-supplied 3x2 integer matrices presented in row major form.

radix1,radix2
: The radix factors for the 3x2 integer matrices.

raw
: If set to TRUE(1) all integer matrices will be in internal (raw) format (i.e. translation values are not scaled).
If set to FALSE(0), all integer matrix values are scaled.

Output Parameters

result
: Resulting 4x4 real matrix obtained by multiplying **matrix1** and **matrix2** .
Resulting 3x2 integer matrix obtained by multiplying **matrix1** and **matrix2**.

radix
: is the resulting radix factor for a 3x2 integer matrix.

Discussion

Concatenation (or matrix multiplication) multiplies **matrix1** and **matrix2** and the resulting matrix is placed in **result**:

matrix1 * matrix2 → result

The three matrices need not be unique. For instance, *concat_matrix* or *intconcat_matrix2d* (A,A,A) is valid.

When multiplying integer matrices they are given an implied third column that is 0,0,1. This allows a full 3x3 matrix multiply, and does not distort the resulting picture. The third column is removed from the result matrix.

Integer transformation matrices are scaled to allow a fractional portion for rotating objects. The radix factor indicates the number of bits to the right of the decimal point. Legal limits are 0 to 30. Once a coordinate has been transformed, it is divided by 2****radix** to return to an integer value.

When using raw mode with an integer matrix positions (2,0) and (2,1) have an implied radix factor of 0(no scaling). This allows large translation ranges along with accurate rotations.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

LANGUAGE DEPENDENCIES

FORTRAN77:

A transposition of array rows and columns is required due to the manner in which FORTRAN77 stores arrays.

SEE ALSO

concat_transformation(3G), flush_matrices(3G), pop_matrix(3G), push_matrix(3G), replace_matrix(3G).

NAME

concat_transformation2d, concat_transformation3d, intconcat_transform2d – pre-concatenate or post-concatenate transformation matrix with *current transformation matrix*

SYNOPSIS

C Syntax:

```
void concat_transformation2d(fildes,xform2,sequence,stack);
int fildes,sequence,stack;
float xform2[3][2];

void concat_transformation3d(fildes,xform3,sequence,stack);
int fildes,sequence,stack;
float xform3[4][4];

void intconcat_transform2d(fildes,xform2,radix,sequence,stack,raw);
int fildes,radix,sequence,stack,raw;
int xform2[3][2];
```

FORTRAN77 Syntax:

(see Language Dependencies below)

```
subroutine concat_transformation2d(fildes,xform2,sequence,stack)
integer*4 fildes,sequence,stack
real xform2(2,3)

subroutine concat_transformation3d(fildes,xform3,sequence,stack)
integer*4 fildes,sequence,stack
real xform3(4,4)

subroutine intconcat_transform2d(fildes,xform2,radix,sequence,stack,raw)
integer*4 fildes,radix,sequence,stack,raw,xform2(2,3)
```

Pascal Syntax:

```
type
        int2d_xform=array[1..3][1..2] of integer;
        two_d_xform = array [1..3][1..2] of real;
        three_d_xform = array [1..4][1..4] of real;

procedure concat_transformation2d(fildes:integer;var xform2:two_d_xform;
        sequence,stack:integer);

procedure concat_transformation3d(fildes:integer;var xform3:three_d_xform;
        sequence,stack:integer);

procedure intconcat_transform2d(fildes:integer;var xform2:int2d_xform;
        radix,sequence,stack,raw:integer);
```

DESCRIPTION

Input Parameters

fildes	Integer file descriptor returned by *gopen* When an I/O path to a graphics device is opened.
xform2	3x2 (2-dimensional transform) matrix.
xform3	4x4 (3-dimensional transform) matrix.
radix	is the radix factor for the 3x2 integer matrix.
sequence	If set to POST, post-concatenation is performed. If set to PRE, pre-concatenation is performed.

stack If set to PUSH, the result matrix is to be pushed on top of the matrix stack.
If set to REPLACE, the result matrix replaces the top matrix on the matrix stack.

raw If set to TRUE(1), the integer matrix is in internal (raw) format (i.e. translation values are not scaled).
If set to FALSE(0), all integer matrix values are scaled.

Discussion

The specified matrix is concatenated with the current transformation matrix (top matrix on the matrix stack). The resulting matrix is pushed on the top of the matrix stack if stack is PUSH. If stack is REPLACE, the resulting matrix replaces (destructive replacement) the current top of matrix stack. In either case, the resulting matrix becomes the *current transformation matrix* and is left on the top of the matrix stack. The current transformation matrix is used to transform subsequent output primitives.

The coordinate systems used by Starbase may be conceptually defined as follows:

1. User points are assumed to be defined in modelling coordinates. These points are transformed into world coordinates by use of a matrix called the *modelling transformation matrix* if any matrices have been pushed on the stack. World coordinates are used to perform any necessary rendering calculations. Use *concat_transformation* to place or modify modelling transformations on the matrix stack after defining the viewing transformation.
2. World coordinates are then transformed to device coordinates by the *viewing transformation matrix*, which is usually the concatenation of user-defined viewing transformations and the vdc-to-device transformation matrix. When a device is opened the *viewing transformation matrix* is simply the vdc-to-device coordinate transformation matrix. *Intview_window, intview_port, intview_matrix, view_camera, view_port, view_window, view_volume*, and *view_matrix* can be used to define further viewing transformations such as perspective.

If a graphics device has been opened in MODEL_XFORM mode, the modelling transformation cannot be combined with the viewing transformation because rendering calculations such as shading may be needed after the modelling-to-world coordinate transformation. For this reason, matrices pushed on the stack are left undisturbed, and all transformations from modelling coordinates to device coordinates occur in two steps: modelling to world coordinates, followed by world to device coordinates.

If a graphics device is not in MODEL_XFORM mode, the modelling and viewing transformations can be combined. Thus the current viewing transformation matrix is post-concatenated to modelling matrices placed on the matrix stack and subsequent output primitives are transformed using only the top matrix on the matrix stack.

If the matrix stack is empty and this function is called with *stack* set to REPLACE, a warning is generated and the matrix stack is left unchanged.

All floating point matrices are maintained in 3-Dimensional (4x4) form. If *concatenate_matrix2d* is called, the elements of *xform2* are mapped to a 4x4 matrix for the concatenation as follows:

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \end{bmatrix} \rightarrow \begin{bmatrix} a_{00} & a_{01} & 0 & 0 \\ a_{10} & a_{11} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a_{20} & a_{21} & 0 & 1 \end{bmatrix}$$

Post-concatenation of matrices should be used very carefully. Unless the top matrix on the matrix stack is a modelling matrix (as in MODEL_XFORM mode), it is not device independent because the post concatenation occurs after the vdc-to-device coordinate transformation. When using this feature, use *inquire_size* to determine the range of device coordinates.

Perspective transformations (as well as any other non-linear or viewing transformation) should not appear in modelling matrices because they distort surface normals and invalidate lighting calculations. They should instead appear in the viewing transformation that was set with *view_transform*. When performing perspective transformations, a perspective model where the eye is at origin of perspective space is recommended. Any other model can easily be modified by a single translation step.

Integer transformation matrices are scaled to allow a fractional portion for rotating objects. The radix factor indicates the number of bits to the right of the decimal point. Legal limits are 0 to 30. Once a coordinate has been transformed, it is divided by 2****radix** to return to an integer value.

When using raw mode with an integer matrix, the positions (a20) and (a21) have an implied radix factor of 0 (no scaling). This allows large translation ranges along with accurate rotations.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

LANGUAGE DEPENDENCIES

FORTRAN77:

FORTRAN requires a transposition of array rows and columns due to the manner in which it stores arrays.

SEE ALSO

concat_matrix(3G), flush_matrices(3G), gopen(3G), inquire_sizes(3G), pop_matrix(3G), push_matrix(3G), replace_matrix(3G), view_transform(3G).

NAME

curve_resolution – set resolution for splines and arcs

SYNOPSIS

C Syntax:

```
void curve_resolution (fildes,coordinate_type,u_interior,v_interior,
                       u_exterior,v_exterior);
int fildes,coordinate_type;
float u_interior,v_interior,u_exterior,v_exterior;
```

FORTRAN77 Syntax:

```
subroutine curve_resolution (fildes,coordinate_type,u_interior,v_interior,
                             u_exterior,v_exterior)
integer*4 fildes,coordinate_type
real u_interior,v_interior,u_exterior,v_exterior
```

Pascal Syntax:

```
procedure curve_resolution (fildes,coordinate_type:integer;
                            u_interior,v_interior,u_exterior,v_exterior:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

coordinate_type Identifies the scale of the resolution value:
DC_VALUES=0, METRIC=1, VDC_VALUES=2, STEP_SIZE=3

u_interior,v_interior,u_exterior,v_exterior
Maximum distance between sample points on spline_curves (u_exterior) and spline_surfaces along interior and exterior boundaries, or the length of elliptical arc chord segments (**u_exterior**).

Discussion

Curve_resolution sets the sample size used between knots in splines, curves, and surfaces; or the chord length of elliptical or circular arc segments. Coordinate types are as follows:

- DC_VALUES is set directly in pixels.
- VDC_VALUES defines a value based on the current *vdc_extent*.
- METRIC assumes the value is in millimeters.
- STEP_SIZE is a value used directly to increment *u* and *v* from zero to their maximum values, or to set a delta radian value for arcs.

If an interior value is specified to be less than the corresponding exterior value, the interior value is set to the exterior value. Interior values are also limited to be no more than a device-dependent factor larger than the exterior values, but such that at least 3 samples result.

DEFAULTS

Exterior values are set to 2 mm and interior values are set to 5 mm.

SEE ALSO

arc(3G), circle(3G), ellipse(3G), spline(3G), *Starbase Graphics Techniques.*

NAME

dbuffer_switch – switch buffers when double buffering

SYNOPSIS

C Syntax:

```
void dbuffer_switch(fildes,buffer);
int fildes,buffer;
```

FORTRAN77 Syntax:

```
subroutine dbuffer_switch(fildes,buffer)
integer *4 fildes,buffer
```

Pascal Syntax:

```
procedure dbuffer_switch(fildes,buffer:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

buffer Least significant bit of this integer determines which buffer (odd or even) to use.

Discussion

dbuffer_switch switches between output buffers when double buffering is enabled. Besides the device file descriptor, this function takes one parameter: **buffer**, the least significant bit of which is used to select the output buffer used when writing to the graphics output device.

When *dbuffer_switch* is called:

1. Starbase waits (if necessary for visual continuity) for the graphics device to enter its vertical retrace period.
2. Either *write_enable/display_enable* or *bank_switch* is used to activate the selected buffer from the double buffer pair.
3. The newly enabled write buffer is cleared using *clear_view_surface*, unless suppressed by the SUPPRESS_CLEAR mode in *double_buffer*.

If double buffering is disabled and *dbuffer_switch* is called, *dbuffer_switch* simply calls *clear_view_surface*, unless suppressed by the SUPPRESS_CLEAR mode in *dbuffer_switch*.

Make_picture_current is not called from *dbuffer_switch*. In general, it need not be called from the application program, since all preceding buffered primitives will be drawn before *dbuffer_switch* is performed.

See the *Starbase Programming with X11* chapter on display control for more information about using this routine with a window system.

DEFAULTS

buffer = 0: Enable lowest buffer for writing.

SEE ALSO

double_buffer (3G), inquire_display_mode (3G), *Starbase Graphics Techniques, Starbase Programming with X11.*

NAME

dc_to_vdc – transform device coordinate point into virtual device coordinate point using the inverse of the current vdc-to-device coordinate transformation.

SYNOPSIS SYNTAX

C Syntax:

```
void dc_to_vdc(fildes,dcx,dcy,dcz,vdcx,vdcy,vdcz)
int fildes;
int dcx,dcy,dcz;
float *vdcx,*vdcy,*vdcz;
```

FORTRAN77 Syntax:

```
subroutine dc_to_vdc(fildes,dcx,dcy,dcz,vdcx,vdcy,vdcz)
integer*4 fildes
integer*4 dcx,dcy,dcz
real vdcx,vdcy,vdcz
```

Pascal Syntax:

```
procedure dc_to_vdc(fildes:integer;dcx,dcy,dcz:integer;
var vdcx,vdcy,vdcz:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

dcx,dcy,dcz Defines a device coordinate point to be transformed by this procedure.

Output Parameters

vdcx,vdcy,vdcz Transformed point in virtual device coordinate values.

Discussion

Dc_to_vdc transforms a point in device coordinates to virtual device coordinate values, using the inverse of the current vdc-to-device coordinates transformation to perform the calculation.

SEE ALSO

vdc_to_dc(3G) transform_point(3G).

NAME

define_color_table – set the color values in the device color table.

SYNOPSIS

C Syntax:

```
void define_color_table(fildes,start,count,colors);
int fildes,start,count;
float colors[][3];
```

FORTRAN77 Syntax:

```
subroutine define_color_table(fildes,start,count,colors)
integer*4 fildes,start,count
real colors(3,count)
```

Pascal Syntax:

```
type
    rgb_color=array[1..3]of real;
procedure define_color_table(fildes,start,count:integer;
    var colors array:[lo..hi:integer] of rgb_color);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

start Integer index into the color table at which to start the color table update.

count Integer number of color table entries to be updated.

colors New red, green, and blue values for each color table entry (floating-point values in the range of 0 to 1, where 1 is full intensity). In C terminology-

```
color_table[start][0]=colors[0][0];  /* red */
color_table[start][1]=colors[0][1];  /* green */
color_table[start][2]=colors[0][2];  /* blue */
 ...
color_table[start+count-1][2] = colors[count-1][2]
```

Discussion

Each file descriptor opened as an output device has a color table. When this procedure is called, the color table and the device's color map are updated. If multiple file descriptors are open to the same device, the color table and the device's color map may not always be identical. For Starbase procedures that have red, green and blue parameters, the closest color is searched for in the color table (not the color map), or it is calculated (according to color map mode set with *shade_mode*).

Some device drivers have a *gescape* (READ_COLOR_MAP) that reads the current color map into the color table. See the appropriate device driver description for more information.

For multibank graphics devices:

16 Planes	Each bank of 8 has the same color map.
24 Planes	If the mode parameter of *shade_mode* is set to CMAP_FULL, when the color table is written, the blue entries are the intensities in bank 1, the green are in bank 2, and the red are in bank 3. Otherwise the device appears to be a standard 8-plane graphics device.

Multibank graphics devices that support video blending may allow separate color map definitions for different banks. See the *Starbase Device Drivers Library Manual* for details.

See the *Starbase Programming with X11* chapter on display control for more information about using this routine with a window system.

If dbuffering, set CMAP in both buffers based on the number of planes returned.

DEFAULTS

All color maps are device-dependent.

SEE ALSO

inquire_color_table(3G), inquire_size(3G), inquire_display_mode(3G), shade_mode(3G), shade_range(3G), *Starbase Device Drivers Library Manual, Starbase Programming with X11.*

NAME

define_raster_echo – define a raster echo to be used on an output device.

SYNOPSIS

C Syntax:

```
void define_raster_echo(fildes,echo_num,dx,dy,posdx,posdy, rule,source);
int fildes,echo_num,dx,dy,posdx,posdy,rule;
unsigned char source[];
```

FORTRAN77 Syntax:

```
subroutine define_raster_echo(fildes,echo_num,dx,dy,
posdx,posdy,rule,source)
integer*4 fildes,echo_num,dx,dy,posdx,posdy,rule
integer*4 source()
```

Pascal Syntax:

```
type gbyte=0..255;

procedure define_raster_echo(fildes,echo_num,dx,dy,
posdx,posdy,rule:integer;var source:packed array[lo..hi] of gbyte);
```

DESCRIPTION

Input Variables

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

echo_num Selects the echo for devices that support more than one simultaneous echo. For single echo devices, use 0 (zero).

dx,dy Rectangle dimensions in pixels.

posdx and **posdy** Define a coordinate point (hot spot) in the raster rectangle that maps to the echo position defined by the x and y offset from the upper left corner.

rule Replacement rule used to combine the echo with the frame buffer. See *drawing_mode* for further information.

source Contains source pixel information.

Discussion

define_raster_echo uses the source pixel information, interpreted as a *dx* by *dy* pixel rectangle, to define the raster echo type #7.

The frame buffer reads the rectangle then combines the raster data with the frame buffer, using the replacement rule specified by *rule*. The saved rectangle is written into the frame buffer when the echo is removed, making a non-destructive echo.

For multi-bank graphics devices that have been set to CMAP_FULL by *shade_mode*, the raster echo can be defined for only one bank at a time. Therefore this function defines the raster echo only for the currently selected bank, as set by *bank_switch*. Raster echoes can be defined for the other banks by selecting them with *bank_switch* and calling *define_raster_echo* again. When CMAP_NORMAL or CMAP_MONOTONIC is set, the raster echo is sent to all banks simultaneously.

When raster echo is enabled, it will be written into all graphics planes by default. See the appropriate device driver description for more information.

The following C program segment shows how to define the raster echo as a pointing finger with the "hot spot" being the finger tip.

```
#define byte unsigned char
static byte hand[16][9]={       0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,
```

```
                                0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,
                                0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,
                                0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,
                                0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,
                                0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,
                                0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,
                                0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,
                                0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0};

        echo_num = 0;

        dx = 16; dy = 9;            /* rectangle dimensions.        */

        posdx = 15; posdy = 3;      /* over 15 down 3 to position
                                       in front of finger tip.      */

        rule = 7;                   /* replacement rule to "or" hand
                                       with contents of frame buffer. */

        define_raster_echo(fildes,echo_num,dx,dy,posdx,posdy,rule,hand);
```

DEFAULTS

Device-dependent default raster definition. See the appropriate device driver description for details.

SEE ALSO

drawing_mode(3G), echo_type(3G), *Starbase Device Drivers Library*.

NAME

depth_cue – enable/disable depth cueing of output primitives

SYNOPSIS

C Syntax:

```
void depth_cue(fildes,on,min);
int fildes, on;
float min;
```

FORTRAN77 Syntax:

```
subroutine depth_cue(fildes, on, min)
integer*4 fildes, on
real min
```

Pascal Syntax:

```
procedure depth_cue(fildes, on:integer; min:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

on If FALSE (0), output primitives are not depth cued.
If TRUE (1), output primitives are depth cued using the current model or the model selected by DC_MIN or DC_COLOR.

The values FALSE and TRUE may be ORed with the following to select the depth cue model to be used. If neither of the following values are specified, or if both values are specified, the current depth cue model is not changed.

DC_MIN output primitives are depth cued using the **min** parameter.
DC_COLOR output primitives are depth cued using the color and reference planes specified by *depth_cue_color* and *depth_cue_range*.

min Minimum intensity for the most distant primitives (range 0.0 to 1.0)

Discussion

Depth cueing modulates the intensity of output primitives proportional to the distance away from the viewer. Primitives at the nearest (front) clipping plane appear at full intensity as set by *line_color*, *fill_color*, etc. When using DC_MIN, primitives at the most distant (rear) clipping plane are dimmer by the factor **min**, with a linear interpolation of values in between.

Depth cueing can also be performed according to a more advanced model (DC_COLOR) by calling *depth_cue_color* and *depth_cue_range*. Calling these routines allows specification of the background color to depth cue towards, and the front and back planes. When the DC_COLOR model is used, the **min** parameter for *depth_cue* is ignored. However, the **on** parameter for *depth_cue* is used to enable or disable depth cueing and select the model.

CMAP_MONOTONIC or CMAP_FULL mode must be set with the *shade_mode* procedure before depth cueing is turned on.

Recommendation

Enable front and back clipping (use *depth_indicator*) when depth cueing is enabled.

DEFAULTS

on = FALSE (0): output primitives drawn without depth cueing.

SEE ALSO

shade_mode (3G), *Starbase Graphics Techniques*.

NAME

depth_cue_color – set color for depth cuing

SYNOPSIS

C Syntax:

```
void depth_cue_color_index(fildes,index);
int fildes,index;

void depth_cue_color(fildes,red,green,blue);
int fildes;
float red,green,blue;
```

FORTRAN77 Syntax:

```
subroutine depth_cue_color_index(fildes,index)
integer*4 fildes,index

subroutine depth_cue_color(fildes,red,green,blue)
integer*4 fildes
real red,green,blue
```

Pascal Syntax:

```
procedure depth_cue_color_index(fildes,index:integer);

procedure depth_cue_color(fildes:integer;red,green,blue:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

index Integer index into the color table. The depth cue color for subsequent primitives which use the color depth cue model will be the color at that location in the color table. Use *inquire_sizes* procedure to find the number of color table entries available for a specific device.

red, green, blue
Color components used for subsequent depth cue calculations. Values range from 0.0 to 1.0 where zero indicates minimum color (off) and 1.0 indicates full (maximum) color. Monochrome devices determine intensity by using the equation:

$$I = 0.30 * red + 0.59 * green + 0.11 * blue$$

Discussion

The color specified is the color at the yon reference plane specified in the *depth_cue_range* procedure.

DEFAULTS

Color Index defaults to 0.

SEE ALSO

depth_cue(3G), depth_cue_range(3G), *Starbase Graphics Techniques*.

NAME

depth_cue_range – set range for depth cuing

SYNOPSIS

C Syntax:

```
void depth_cue_range(fildes,hither,yon,hither_scale,yon_scale);
int fildes;
float hither,yon,hither_scale,yon_scale;
```

FORTRAN77 Syntax:

```
subroutine depth_cue_range(fildes,hither,yon,hither_scale,yon_scale)
integer*4 fildes;
real hither,yon,hither_scale,yon_scale
```

Pascal Syntax:

```
procedure depth_cue_range(fildes:integer;hither,yon,hither_scale,yon_scale:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

hither,yon define the reference planes, parallel to the XY plane, at which the depth cue scaling factors are specified. These values are in perspective vdc coordinates; i.e. the perspective divide occurs after the depth cue calculation.

hither_scale,yon_scale
define the weights that determine how a primitive's color is combined with the depth cue color.

Discussion

When primitives are depth cued using the color depth cue model, colors are interpolated toward the color specified in *depth_cue_color*. The color of a point in between the hither depth cue plane and the yon depth cue plane is determined by a linear interpolation of the hither_scale and yon_scale factors. Any points in front of the depth cue hither plane or behind the depth cue yon plane are rendered in a constant color. The applicable equations:

If Z is in front of depth cue hither:

Color = hither_scale * primitive_color + (1 − hither_scale) * depth_cue_color

If Z is in back of depth cue yon:

Color = yon_scale * primitive_color + (1 − yon_scale) * depth_cue_color

If Z is in between depth cue hither and depth cue yon:

Color = r * primitive_color + (1 − r) * depth_cue_color

Where:

r = yon_scale + ((Z − yon) * (hither_scale − yon_scale)) / (hither − yon)

DEFAULTS

hither = 0.0; yon = 1.0; hither_scale = 1.0; yon_scale = 0.0

SEE ALSO

depth_cue(3G),depth_cue_color(3G), *Starbase Graphics Techniques*.

NAME

depth_indicator – enable/disable clipping to front and back clipping planes.

SYNOPSIS

C Syntax:

```
void depth_indicator (fildes,front_on,back_on);
int fildes, front_on,back_on;
```

FORTRAN77 Syntax:

```
subroutine depth_indicator(fildes,front_on,back_on)
integer*4 fildes,front_on,back_on
```

Pascal Syntax:

```
procedure depth_indicator(fildes,front_on,back_on:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when the I/O path to the output graphics device is opened.

front_on Enables front-clipping-plane clipping if set to TRUE (1); disables front-plane clipping if set to FALSE (0).

back_on Enables back-clipping-plane clipping if set to TRUE (1); disables back-plane clipping if set to FALSE (0).

Discussion

Depth_indicator enables/disables front clipping plane, and/or back clipping plane. Clipping to the front plane should only be disabled when the primitives are known to be beyond the front plane. Otherwise, strange images may result from objects passing through the focal point of a perspective view.

3-dimensional processing speed is considerably better if front and back clipping are disabled.

DEFAULTS

front_on = TRUE, **back_on** = FALSE.

SEE ALSO

clip_depth(3G), clip_indicator(3G), clip_rectangle(3G).

NAME

designate_character_set – associate a G-set with a character set.

SYNOPSIS

C Syntax:

```
void designate_character_set(fildes,chset,gset);
int fildes,gset;
char *chset;
```

FORTRAN77 Syntax:

```
subroutine designate_character_set(fildes,chset,gset)
integer*4 fildes,gset
character*(*) chset
```

Pascal Syntax:

```
procedure designate_character_set(fildes:integer;
chset:string255;gset:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

chset Name of a character set's directory.

gset Selects one of the four G-sets (0 - 3). G-sets are discussed in the *Starbase Graphics Techniques* manual.

Discussion

Designate_character_set inserts the string **'/usr/lib/starbase/stroke'** in front of the character set directory name specified by the **chset** parameter. The resulting path name is used to identify the character set to be associated with the G-set specified by the **gset** parameter. The actual path name for the font to be used is the concatenation of the character set path name and the font index. The font index is specified by the *text_font_index* procedure. This implies that the character set path name is a directory.

The following character sets are supported by Starbase:

chset	Name	Pathname
usascii	USASCII	/usr/lib/starbase/stroke/usascii
jisascii	JISASCII	/usr/lib/starbase/stroke/jisascii
katakana	Katakana	/usr/lib/starbase/stroke/katakana
hproman	HP Roman	/usr/lib/starbase/stroke/hproman
kanji	Kanji	/usr/lib/starbase/stroke/kanji
jpn	Jpn	/usr/lib/starbase/stroke/jpn

When the text switching mode is HP_8BIT, only G0 and G1 can be set.
The *kanji* character set is available from Hewlett-Packard as a separate product.

DEFAULTS

G0, G2 USASCII (as set in */usr/lib/starbase/defaults*).
G1, G3 HP ROMAN (as set in */usr/lib/starbase/defaults*).

SEE ALSO

text_font_index(3G), text_switching_mode(3G), *Starbase Graphics Techniques*.

NAME

disable_events – disable events queuing from specified graphics input device

SYNOPSIS

C Syntax:

```
int disable_events(fildes,class,ordinal);
int fildes,class,ordinal;
```

FORTRAN77 Syntax:

```
integer*4 function disable_events(fildes,class,ordinal)
integer*4 fildes,class,ordinal
```

Pascal Syntax:

```
function disable_events(fildes,class,
        ordinal:integer):integer;
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

class One of the following device classes: LOCATOR, CHOICE or ALL. The integer values assigned for these class designators are found in the Starbase **include** file used with your program.

ordinal Logical device number ranging from 1 to the number of devices in the specified class. LOCATOR and CHOICE devices have device-dependent ordinal values. See *inquire_input_capabilities* for details. If **class** equals ALL, then **ordinal** is ignored.

Discussion

Disable_events disables the queuing of events in the Starbase event queue from the device specified by **fildes**, **class**, and **ordinal**. A single event queue is used by all system graphics input devices. The function returns 0 when successful.

If **class** is set to ALL, **ordinal** is ignored and events are disabled for all logical devices specified by **fildes**.

If the device is not initialized or does not exist, a non-zero function value is returned. If the device was not enabled for events, nothing is done and the return value is 0, indicating success.

This call does **not** cause flushing of the event queue.

DEFAULTS

All devices are initially disabled for event generation.

SEE ALSO

await_event(3G), enable_events(3G), gopen(3G), inquire_input_capabilities(3G), set_signals(3G).

NAME

display_enable – select which planes of a raster device are to be displayed

SYNOPSIS

C Syntax:

```
void display_enable(fildes,plane_mask);
int fildes,plane_mask;
```

FORTRAN77 Syntax:

```
subroutine display_enable(fildes,plane_mask)
integer*4 fildes,plane_mask
```

Pascal Syntax:

```
procedure display_enable(fildes,plane_mask:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

plane_mask Mask used to determine which planes are enabled to drive the visual output. Each bit is interpreted as the plane enable of that respective plane; i.e., If least significant bit is 0 then plane 1 is disabled. If 2nd bit is 1, then plane 2 is enabled, etc.

Discussion

Disabled frame buffer device planes drive the visual output as if they were zeros. For example, if the *background_color_index* is set to 170 (hex AA) and the display is cleared then: for **plane_mask** = 255 (hex FF), the color seen is the color at color table index 170. For **plane_mask** = 15 (hex 0F), the color seen is the color at color table index 10 (hex 0A), etc. This procedure, along with *write_enable* is very useful for writing into undisplayed bit planes while leaving the displayed planes unaltered (double-buffering).

For multi-bank graphics devices **plane_mask** is a byte quantity that affects the planes of all displayed banks simultaneously. Graphics devices which support video blending may treat the **plane_mask** as a 24 bit quantity while blending is enabled. See the *Starbase Device Drivers Library Manual* for details.

Double-buffering with 12 planes or with fewer than 6 planes uses *display_enable* and *write_enable* to select buffers (other values use *bank_switch*). During double buffering, *display_enable* and *write_enable* are applied to only the displayed and written planes as determined by *double_buffer*.

Some devices may not support this operation. See the *Starbase Programming with X11* chapter on display control for more information about using this routine with a window system.

DEFAULTS

plane_index = 255: All planes (in first bank, if multi-bank) enabled.

SEE ALSO

write_enable(3G), *Starbase Device Drivers Library Manual, Starbase Programming with X11.*

NAME

double_buffer – enable/disable double buffering

SYNOPSIS

C Syntax:

```
int double_buffer(fildes,mode,planes);
int fildes, mode,planes;
```

FORTRAN77 Syntax:

```
integer*4 function double_buffer(fildes,mode,planes)
integer*4 fildes, mode, planes
```

Pascal Syntax:

```
function double_buffer(fildes,mode,planes:integer):integer;
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

mode Specifies whether double buffering is on or off, whether to update the color map and whether to draw to the same buffer that is displayed.

planes Specifies the number of planes in each buffer for input. Allowable requests are 1, 2, 3, 4, 6, 8 and 12.

Discussion

double_buffer is used to enable/disable double buffering and define buffering parameters. In addition to the device file descriptor, this function accepts two parameters: **mode** and **planes**.

- If the least significant bit of **mode** is set to TRUE (1), double buffering is enabled and the display and write enable registers are set up according to the hardware configuration and **planes** parameter. In double buffer mode, the write enable and display enable masks as set by *write_enable* and *display_enable* are applied individually to each of the buffers. For example, if bit 1 of the write enable mask is clear, then plane 1 of both buffers will be disabled for writing. If INIT is ORed with **mode**, the color map is set up properly and color indices are correctly interpreted. If DFRONT is ORed with **mode**, when **mode** is set to TRUE (1), then the buffer that is enabled for display will also be enabled for writing. This makes it much easier to effectively switch in and out of double buffering. If SUPPRESS_CLEAR is ORed with **mode**, then the buffer that is enabled for writing will not be cleared by subsequent calls to *dbuffer_switch*.
- The **planes** parameter specifies the number of planes to be used for each buffer. Legal values for this parameter are 1, 2, 3, 4, 6, 8, or 12. The number of planes actually being used for double buffering is returned by this function. Obviously an application must not request more than half the number of planes installed on the graphics device being used. If this occurs, Starbase sets the number of planes used to the maximum legal value. Conversely, if an application requests fewer than the maximum number of planes, the uppermost planes are used. Unused planes are write disabled and display enabled. Note that if fewer than 3 planes are requested for each buffer, the color map mode should be set to CMAP_NORMAL or CMAP_MONOTONIC (see SHADE_MODE).

After double buffering has been set up with *double_buffer*, the buffers can be switched using *dbuffer_switch*.

If only one plane is available on the device or the color map mode is CMAP_FULL and there are fewer than 6 planes available on the device, double buffering cannot be enabled and an error is generated, if requested. Subsequent calls to *dbuffer_switch* result in only a *clear_view_surface*. The *clear_view_surface* will be suppressed if SUPPRESS_CLEAR was ORed with **mode** in the call to *double_buffer*.

When double buffering is disabled, Starbase returns the display and write enable registers to their current values and/or resets the *bank_select* to its current value.

If 6 planes are requested when 8 are available, all 8 will actually be used.

Double buffering 12 planes is only possible in CMAP_FULL mode. 8 or less planes will be used in other color map modes. *double_buffer* does not set *dbuffer_switch*.

RETURN VALUE

Upon completion of this function, a non-negative number is returned that is the actual number of planes being used for double buffering. This number is the number requested unless the device cannot support that many planes.

DEFAULTS

mode = 0: double buffering off.

SEE ALSO

dbuffer_switch (3G), write_enable (3G), *Starbase Graphics Techniques.*

NAME

draw2d, draw3d, dcdraw, intdraw2d – draw line from current to specified pen position

SYNOPSIS

C Syntax:

```
void draw2d(fildes,x,y)
int fildes;
float x,y;

void draw3d(fildes,x,y,z)
int fildes;
float x,y,z;

void dcdraw(fildes,x,y)
int fildes,x,y;

void intdraw2d(fildes,x,y)
int fildes,x,y;
```

FORTRAN77 Syntax:

```
subroutine draw2d(fildes,x,y)
integer*4 fildes
real x,y

subroutine draw3d(fildes,x,y,z)
integer*4 fildes
real x,y,z

subroutine dcdraw(fildes,x,y)
integer*4 fildes,x,y

subroutine intdraw2d(fildes,x,y)
integer*4 fildes,x,y
```

Pascal Syntax:

```
procedure draw2d(fildes:integer;x,y:real);

procedure draw3d(fildes:integer;x,y,z:real);

procedure dcdraw(fildes,x,y:integer);

procedure intdraw2d(fildes,x,y:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when the I/O path to the output graphic device is opened.

x, y, z Defines the position to draw to and the new *current pen position*.

Discussion

All drawing is performed using the current line attributes, *drawing_mode* and *write_enable*.

Procedures *draw2d*, *draw3d*, and *intdraw2d* use world coordinate values, while *dcdraw* uses device coordinate values.

Dcdraw draws a line in device coordinates without going through any transformations or clipping steps. *Dcdraw* is device-dependent and should **not** be used when portability between devices is desirable.

If a series of draws and/or moves are to be executed in succession, the *polyline* procedure is much faster.

The current pen position for device coordinate operations is entirely different than that of world coordinates. Device coordinate current pen position is only consistent between device coordinate operations. After any world coordinates operation, a *dcmove* should be performed before any other device coordinate operations.

Similarly, the world coordinate current pen position is only consistent between world coordinate operations.

Note that when using a line style other than solid, the pattern may not be continuous between draws. The pattern may restart at the beginning of any draw. If a continuous pattern is important, use *polyline*. Line style is not supported on lines wider than zero width.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

For increased performance for *draw2d* and *draw3d*, macros can be used to generate in-line code for C. See "Move and Draw Macros" in *Starbase Graphics Techniques – HP-UX Concepts and Tutorials*.

SEE ALSO

drawing_mode(3G), inquire_current_position(3G), line_color(3G), line_endpoint(3G), line_repeat_length(3G), line_type(3G), line_width(3G), move(3G), polyline(3G), write_enable(3G), *Starbase Graphics Techniques*.

NAME

drawing_mode – select the pixel replacement rules for subsequent raster operations and output primitives.

SYNOPSIS

C Syntax:

```
void drawing_mode(fildes,replacement_rule);
int fildes,replacement_rule;
```

FORTRAN77 Syntax:

```
subroutine drawing_mode(fildes,replacement_rule)
integer*4 fildes,replacement_rule
```

Pascal Syntax:

```
procedure drawing_mode(fildes,replacement_rule:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

replacement_rule

Defines the combination of source and destination pixel values:

Replacement Rules	Resulting Destination Value
0	ZERO
1	Source AND Destination
2	Source AND NOT Destination
3	Source (the default rule)
4	NOT Source AND Destination
5	Destination
6	Source EXCLUSIVE OR Destination
7	Source OR Destination
8	NOT source AND NOT Destination
9	Source EXCLUSIVE NOR Destination
10	NOT Destination
11	Source OR NOT Destination
12	NOT Source
13	NOT Source OR Destination
14	NOT Source OR NOT Destination
15	ONE

Discussion

Drawing_mode sets the replacement rule used by both the raster operation functions and all the output primitives.

This procedure only applies to raster devices. Some raster devices may not support all drawing modes.

DEFAULTS

Replacement_rule = 3 (source).

SEE ALSO

Starbase Device Drivers Library.

NAME

echo_type, dcecho_type, intecho_type2d – change the type of echo being used on an output device.

SYNOPSIS

C Syntax:

```
void echo_type(fildes,echo_number,echo_value,x,y,z);
int fildes,echo_number,echo_value;
float x,y,z;

void dcecho_type(fildes,echo_number,echo_value,dcx,dcy);
int fildes,echo_number,echo_value,dcx,dcy;

void intecho_type2d(fildes,echo_number,echo_value,x,y);
int fildes,echo_number,echo_value,x,y;
```

FORTRAN77 Syntax:

```
subroutine echo_type(fildes,echo_number,echo_value,x,y,z)
integer*4 fildes,echo_number,echo_value
real x,y,z

subroutine dcecho_type(fildes,echo_number,echo_value,dcx,dcy)
integer*4 fildes,echo_number,echo_value,dcx,dcy

subroutine intecho_type2d(fildes,echo_number,echo_value,x,y)
integer*4 fildes,echo_number,echo_value,x,y
```

Pascal Syntax:

```
procedure echo_type(fildes,echo_number,echo_value:integer;
        x,y,z:real);

procedure dcecho_type(fildes,echo_number,echo_value,dcx,dcy:integer);

procedure intecho_type2d(fildes,echo_number,echo_value,x,y:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

echo_number Selects a particular echo to be modified if the graphic device supports more than one simultaneous echo. For devices with single echo outputs, use 0 (zero).

echo_value Type of echo to be used by the device. Predefined types include:

0	No echo
1	Device's best echo
2	Full screen cross hair
3	Small tracking cross
4	Rubber band line
5	Rubber band rectangle
6	Alpha-digital representation
7	User-defined raster cursor
>= 8	Device-dependent representation

There is no maximum value for **echo_value**, so if a device does not support any of the above echoes, **1** is used.

x,y,z Define echo position in virtual device coordinates.

dcx,dcy Define echo position in device coordinates.

Discussion

Echo_type defines what type of echo is to be used for the echo number specified and the echo's initial location. For echo types that use one stationary point and one moving point (such as rubber-band-line echo), this initial position is also used as a stationary (base) point.

The echo type of each echo associated with an output device is initialized to zero (0) by **gopen**.

The performance of output primitives for most devices is reduced if echo is enabled, so echo should usually be disabled when it is not needed.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

SEE ALSO

define_raster_echo(3G), echo_update(3G).

NAME

echo_update, dcecho_update, intecho_update2d – change output device's echo position

SYNOPSIS

C Syntax:

```
void echo_update(fildes,echo_number,x,y,z);
int fildes,echo_number;
float x,y,z;

void dcecho_update(fildes,echo_number,dcx,dcy);
int fildes,echo_number,dcx,dcy;

void intecho_update2d(fildes,echo_number,x,y);
int fildes,echo_number,x,y;
```

FORTRAN77 Syntax:

```
subroutine echo_update(fildes,echo_number,x,y,z)
integer*4 fildes,echo_number
real x,y,z

subroutine dcecho_update(fildes,echo_number,dcx,dcy)
integer*4 fildes,echo_number,dcx,dcy

subroutine intecho_update2d(fildes,echo_number,x,y)
integer*4 fildes,echo_number,x,y
```

Pascal Syntax:

```
procedure echo_update(fildes,echo_number:integer;
        x,y,z:real);

procedure dcecho_update(fildes,echo_number,dcx,dcy:integer);

procedure intecho_update2d(fildes,echo_number,x,y:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

echo_number Selects the particular echo to be moved if the graphics device supports more than one simultaneous echo. For single-echo devices, use 0 (zero).

x,y,z Defines the updated echo position in virtual device coordinates.

dcx,dcy Defines the updated echo position in device coordinates.

Discussion

Echo_update, *dcecho_update*, and *intecho_update2d* define an updated echo position for the echo number specified. For echo types that use one stationary point and one moving point determined by the input device's location, such as rubber-band-line echo, this updated position is used as the moving point. The initial position set with the *echo_type* command is then used as the base point of the echo.

If tracking is on, *echo_update* has little effect on echo position because the tracking process is constantly moving it to the current locator position.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

SEE ALSO

define_raster_echo(3G), echo_type(3G), set_locator(3G), track(3G).

NAME

ellipse, partial_ellipse - define an elliptical region to be filled and/or edged.

SYNOPSIS

C Syntax:

```
void ellipse (fildes,x_radius,y_radius,
              x_center,y_center,rotation);
int fildes;
float x_radius,y_radius,x_center,y_center,rotation;

void partial_ellipse (fildes,x_radius,y_radius,
              x_center,y_center,rotation,closure);
int fildes,closure;
float x_radius,y_radius,x_center,y_center,rotation;
```

FORTRAN77 Syntax:

```
subroutine ellipse (fildes,x_radius,y_radius,x_center,y_center,rotation)
integer*4 fildes
real x_radius,y_radius,x_center,y_center,rotation

subroutine partial_ellipse (fildes,x_radius,y_radius,x_center,y_center,
              rotation,close_type,closure)
integer*4 fildes,close_type
real x_radius,y_radius,x_center,y_center,rotation
```

Pascal Syntax:

```
procedure ellipse (fildes:integer; x_radius,y_radius,
              x_center,y_center,rotation:real);

procedure partial_ellipse (fildes:integer;
              x_radius,y_radius,x_center, y_center,rotation:real;
              closure:integer);
```

DESCRIPTION

Input Parameters

fildes — Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

x_radius,y_radius — Radii of the ellipse in the *x* and *y* directions respectively.

x_center,y_center — Coordinate of the ellipse center.

rotation — Specifies the amount the ellipse is to be rotated in radians.

closure — If TRUE (1), and the polygon vertex list is non-empty and the last subpolygon was not yet closed, a boundary is added to close that subpolygon with a boundary. If FALSE (0), no boundary is added.

Discussion

An ellipse of specified radii centered at (**x_center,y_center**) is filled and/or outlined according to the current interior style. As with all output primitives, the arc is affected by the current drawing mode and write enable. The curve is drawn with chords of length specified in *curve_resolution*. For *partial_ellipse*, the vertices of the arc are added to the polygon vertex list, as in the case of *partial_polygons*.

If the transform mode has been set to THREE_D, the *z* value used is that of the current position. Circles are drawn by specifying **x_radius** and **y_radius** to be equal. Non-filled ellipses are generated by setting the interior style to INT_HOLLOW and edged.

NOTE

Doing any other Starbase call besides *partial_arc*, *partial_ellipse*, *partial_polygon2d*,

partial_polygon3d, or *dcpartial_polygon*, in the middle of a list of procedure calls that add vertices to the polygon buffer and before a call to *polygon*, *rectangle*, *arc*, or *ellipse* produces unpredictable, device-dependent results.

SEE ALSO

arc(3G), circle(3G), curve_resolution(3G), drawing_mode(3G), fill_color(3G), interior_style(3G), perimeter_color(3G), perimeter_type(3G), perimeter_repeat_length(3G), polygon(3G), write_enable(3G), *Starbase Graphics Techniques*.

NAME

enable_events – enable queuing of events from the named input device

SYNOPSIS

C Syntax:

```
void enable_events(fildes,class,ordinal);
int fildes,class,ordinal;
```

FORTRAN77 Syntax:

```
subroutine enable_events(fildes,class,ordinal)
integer*4 fildes,class,ordinal
```

Pascal Syntax:

```
procedure enable_events(fildes,class,ordinal:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

class One of the following device classes: LOCATOR, CHOICE, or ALL. The integer values assigned to these class designators are found in the Starbase **include** file used with your programs.

ordinal Logical device number ranging from 1 to the number of devices in the specified class. LOCATOR and CHOICE devices have device-dependent ordinal values. See *inquire_input_capabilities* for details. If **class** equals ALL, **ordinal** is ignored.

Discussion

Enable_events enables the queuing of events into the event queue from the device specified by **fildes**, **class**, and **ordinal**. A single event queue is used by all system graphics input devices.

If **class** is set to ALL, **ordinal** is ignored and events are enabled for all logical devices of this **fildes**.

When any event occurs, an event report is placed in the common event queue and appropriate signals sent. The event queue can be examined by using the *await_event* call.

Any resources needed to enable and generate events are acquired or allocated at this time. Enable events on a logical device does not affect the state of other devices on the same physical device.

DEFAULTS

All devices are initially disabled for event generation.

SEE ALSO

await_event(3G), disable_event(3G), set_signals(3G).

NAME

file_print – print bitmapfile contents on a raster printer

SYNOPSIS

C Syntax:

```
void file_print(source,formatter,config,print_mode,
                rotate,foreground,background,noback)
char *source,*formatter,*config;
int print_mode;
int rotate,foreground,background,noback;
```

FORTRAN77 Syntax:

```
subroutine file_print(source,formatter, config,
                print_mode, rotate,foreground,background,noback)
character*(*) source,formatter,config
integer*4 print_mode
integer*4 rotate,foreground
integer*4 background,noback
```

Pascal Syntax:

```
procedure file_print(source,formatter,config:string255;
                print_mode,rotate,foreground,background,
                noback:integer);
```

DESCRIPTION

Input Parameters

source Pathname of a raster file containing data to be printed.

formatter Character representation of the formatter to be selected. See *Starbase Device Drivers Library* (formatter sections) for details.

config Pathname of a configuration file where information pertaining to the raster formatter and output device will be found. The contents of this file are formatter-dependent.

print_mode Integer value indicating how the data is to be read. If **print_mode** is non-negative, it specifies a single plane to be printed. If it exceeds the number of planes available, a failure flag is returned. If *print_mode* is negative or equals ALL_PLANES, all planes in the file are printed.

rotate If TRUE (1), indicates that the formatter should rotate the data 90 degrees for output. The direction of the rotation is formatter-dependent. If FALSE (0), the formatter-dependent default orientation is used.

foreground, background
Color map indices to be used when printing a single plane. The color corresponding to **foreground** is printed for **1**s in the source, and the color corresponding to **background** is printed for **0**s (however, see the **noback** parameter, below).

noback If TRUE (1), the color map index specified by **background** is mapped to no color; that is, it is not printed. This parameter has effect whether a single plane or all planes are being printed. If FALSE (0), the background color in the file data (or the specified background color if a single plane) is printed just as any other color.

Discussion

The entire extent of the rectangle recorded in the file is printed, subject to clipping by the formatter to fit the output medium. RGB information is converted by the formatter (through dithering) according to the capabilities of the output device. Actual data written is controlled by the formatter and the configuration file.

If a color map was stored as part of the file, it is used to print the data. If no color map was stored, a default is used based on the color map mode recorded in the file: a CMAP_NORMAL file uses the equivalent of the Starbase default software color map; a CMAP_MONOTONIC or CMAP_FULL file uses the default color map that *shade_mode* would initialize for the same number of planes if double-buffering was disabled at the time.

Conditions under which the operation fails include specification of an invalid formatter string (one not found in the formatter table linked with the program) and conditions under which the formatter itself returns a failure indication. *Inquire_gerror*(3G) can be used to determine the error.

SEE ALSO

bitmap_print(3G), *Starbase Device Drivers Library Manual.*

NAME

file_to_bitmap, file_to_dcbitmap, file_to_intbitmap – copy bitmapfile contents into a bitmap

SYNOPSIS

C Syntax:

```
void file_to_bitmap(fildes,full_depth,spn,dpn,source,xstart,ystart,
        update_cmap)
int fildes,full_depth,spn,dpn;
char *source;
float xstart,ystart;
int update_cmap;

void file_to_dcbitmap(fildes,full_depth,spn,dpn,source,xstart,ystart,
        update_cmap)
int fildes,full_depth,spn,dpn;
char *source;
int xstart,ystart,update_cmap;

void file_to_intbitmap(fildes,full_depth,spn,dpn,source,xstart,ystart,
        update_cmap)
int fildes,full_depth,spn,dpn;
char *source;
int xstart,ystart,update_cmap;
```

FORTRAN77 Syntax:

```
subroutine file_to_bitmap(fildes,full_depth,spn,dpn,source,xstart,ystart,
        update_cmap)
integer*4 fildes,full_depth,spn,dpn
character*(*) source
integer*4 xstart,ystart,update_cmap

subroutine file_to_dcbitmap(fildes,full_depth,spn,dpn,source,xstart,
        ystart,update_cmap)
integer*4 fildes,full_depth,spn,dpn
character*(*) source
integer*4 xstart,ystart,update_cmap

subroutine file_to_intbitmap(fildes,full_depth,spn,dpn,source,xstart,
        ystart,update_cmap)
integer*4 fildes,full_depth,spn,dpn
character*(*) source
integer*4 xstart,ystart,update_cmap
```

Pascal Syntax:

```
procedure file_to_bitmap(fildes,full_depth,spn,dpn:integer;
        source:string255;xstart,ystart:real;
        update_cmap:integer);

procedure file_to_dcbitmap(fildes,full_depth,spn,dpn:integer;
        source:string255;xstart,ystart,
        update_cmap:integer);

procedure file_to_intbitmap(fildes,full_depth,spn,dpn:integer;
        source:string255;xstart,ystart,update_cmap:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when the I/O path of a bitmap device is opened.

full_depth Indicates what source data is to be written into the bitmap. If TRUE (1), all file planes are transferred to the bitmap (**spn** and **dpn** parameters are ignored). If FALSE (0), the **spn** paramter determines a single plane to be transferred.

spn Number of a single plane to copy from the file. It must be in the range 0 to $n-1$, where n is the number of planes in the file.

dpn Destination plane number in the bitmap. **dpn** must be in the range 0 to $m-1$, where m is the number of planes in the bitmap.

source Pathname of the file where the raster information is stored.

xstart, ystart Coordinates of the upper left corner of the destination rectangle in virtual device coordinates (*file_to_bitmap*, *file_to_intbitmap*) or device coordinates (*file_to_dcbitmap*).

update_cmap If TRUE (1), indicates that the software and hardware color maps are updated if there is a color table stored in the file. If FALSE (0), or if there is not a color table in the file, the current color maps are not changed.

Discussion

For the following discussion, pixel data is considered to be an **xlen** by **ylen** pixel rectangle, where **xlen** and **ylen** are the respective values contained in the bitmap file. The pixel data from the source is written into the destination, beginning at **xstart,ystart** according to the current drawing mode and write enable.

The actual rectangle written to the bitmap depends upon the size of the bitmap in the file and the current clipping mode. The source rectangle is clipped to the current clipping rectangle and copied into the destination rectangle.

For full-depth operation with multi-bank graphics devices, the currently selected bank (as set by *bank_switch*) is written. If the current shade mode is CMAP_FULL and the file contains full mode data, all appropriate banks will be written.

Conditions that might cause failure include: file not found, improper file permissions, file does not contain raster data, and requested plane does not exist.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

SEE ALSO

bitmap_to_file(3G), inquire_file(3G), file_print(3G).

NAME

fill_color, bf_fill_color – set color table index for filled areas on raster devices; set the fill color for backfacing polygons.

SYNOPSIS

C Syntax:

```
void fill_color_index(fildes,index);
int fildes,index;

void fill_color(fildes,red,green,blue);
int fildes;
float red,green,blue;

void bf_fill_color_index(fildes,index);
int fildes,index;

void bf_fill_color(fildes,red,green,blue);
int fildes;
float red,green,blue;
```

FORTRAN77 Syntax:

```
subroutine fill_color_index(fildes,index)
integer*4 fildes,index

subroutine fill_color(fildes,red,green,blue)
integer*4 fildes
real red,green,blue

subroutine bf_fill_color_index(fildes,index)
integer*4 fildes,index

subroutine bf_fill_color(fildes,red,green,blue)
integer*4 fildes
real red,green,blue
```

Pascal Syntax:

```
procedure fill_color_index(fildes,index:integer);

procedure fill_color(fildes:integer;red,green,blue:real);

procedure bf_fill_color_index(fildes,index:integer);

procedure bf_fill_color(fildes:integer;red,green,blue:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

index Integer index into the color table. The fill color for subsequent fill primitives, including backfacing elements, will be the color at that location in the color table. Use *inquire_sizes* procedure to find the number of color table entries available for a specific device.

red, green, blue

Color components used for backfacing, as well as non-backfacing, elements of subsequent filled areas. Values range from 0.0 to 1.0 where zero indicates minimum color (off) and 1.0 indicates full (maximum) color. Monochrome devices determine intensity by using the equation:

$$I = 0.30 * red + 0.59 * green + 0.11 * blue$$

Discussion

Polygons and rectangles are filled by the specified color when *interior_style* is INT_SOLID. Backfacing polygons use the specified fill color when *bf_interior_style is* INT_SOLID, INT_OUTLINE, or INT_POINT, and *bf_control* has been called with the **attr** parameter set to TRUE(1).

The color of polygon primitives and backfacing polygons depends on:

- The state of the shading parameter as specified by *shade_mode* which determines whether or not polygons are sent through light source equations.
- **Coord**, **use**, and **normals** parameters as specified by *vertex_format* which determines whether or not to use color coordinates supplied with each vertex.

If **shading** is FALSE and **use** is 0:

When the fill color or backface fill color is set by use of red-green-blue parameters, determination of actual output depends on the state of the mode parameter set with *shade_mode*:

CMAP_NORMAL — This is the default mode if *shade_mode* has not been called. If *colors* (specified by *fill_dither*) is 1, the color table is searched for the closest approximation to the specified color when *fill_color* or *bf_fill_color* is called. When the closest color has been found, the corresponding index value is placed in each of the 16 pixel color index locations in the dither cell. If *colors* (specified by *fill_dither*) is 2, the color table is searched to find two indexes that, when combined in a 50% mixture, will be closest to the color specified when *fill_color* or *bf_fill_color* is called. Each index value is then copied into eight pixel locations in the dither cell. If *colors* is 16, the color table is searched for a combination of 16 indexes that, when combined, come closest to the color requested. In this case, it is unlikely that 16 different index values would be used to produce the specified color. Note: Each color map search takes time. Therefore, the more colors requested in *fill_dither*, the longer each *fill_color* or *bf_fill_color* call will take.

CMAP_MONOTONIC — The color values are converted to an intensity value by using the equation:

$$I = 0.30 * \text{red} + 0.59 * \text{green} + 0.11 * \text{blue}$$

This intensity is mapped to an index, using the minimum and maximum limits defined by *shade_range*. If the intensity does not map to an exact index value, dithering is used between two adjacent indices if *colors* (specified by *fill_dither*) is greater than one.

CMAP_FULL — Color values are mapped directly to an index with the assumption that the color map is set up to a predefined full-color state. If the colors do not map to a exact index, dithering is used if *colors* (specified by *fill_dither*) is greater than one.

When in CMAP_NORMAL mode, *fill_color_index* is faster than *fill_color* and *bf_fill_color_index* is faster than *bf_fill_color* because less time is spent searching for an optimum color match in each dither cell.

Search time can also be shortened by using fewer colors in a dither cell to further shorten color table search time. However, the limited color resolution makes it more difficult to match a given specified color.

For **use** >0, **coord** equal to 3, and **normals** FALSE, *fill_color* is ignored and the polygon color is determined by extra coordinates given with each vertex. See *vertex_format* for more information.

For **shading** TRUE, *fill_color* or *bf_fill_color* is used to set the reflectance coefficient for the light source equations (if no rgb values are provided for each vertex). If an index is given, the color given by the corresponding color table location is used as the reflectance coefficient. See *light_model* for information about light source equations.

The color of backfacing polygons is also set by the *backface_control* procedure. The last color specified by either *backface_control* or *bf_fill_color* will be the color used.

DEFAULTS

Color Index as well as Backface Color Index defaults to 1, i.e., device-dependent dithering for colors not specified with an index.

SEE ALSO

backface_control(3G), bf_control(3G), define_color_table(3G), fill_dither(3G), interior_style(3G), light_model(3G), polygon(3G), rectangle(3G), shade_mode(3G), vertex_format(3G), *Starbase Graphics Techniques.*

NAME

fill_dither – set color value for filled areas on raster devices.

SYNOPSIS

C Syntax:

```
void fill_dither(fildes,colors);
int fildes,colors;
```

FORTRAN77 Syntax:

```
subroutine fill_dither(fildes,colors)
integer*4 fildes,colors
```

Pascal Syntax:

```
procedure fill_dither(fildes,colors:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

colors Number of colors to be searched for and placed in the dither cell. Only 1, 2, 4, 8 and 16 are allowed. Devices may limit the number of options implemented for this parameter, depending on device capabilities.

Discussion

Dithering is a method of generating more fill colors than are available in the color table. It uses the human eye's ability to integrate or blend closely-spaced small pinpoints of color called pixels, each of which can be set to certain predefined colors, into an average color over the entire dither cell. Each dither cell is composed of 16 (red,blue,green) pixel triples where each triple can be set to certain combinations of color and relative intensity in order to produce the desired dither cell color. For a more extensive explanation of how dithering is achieved, see *Starbase Graphics Techniques*.

See *fill_color* for an in-depth discussion of how *fill_color* and *fill_dither* are interrelated.

DEFAULTS

Device-dependent dithering for colors specified with *fill_color*, but not *fill_color_index*.

SEE ALSO

backface_control(3G), define_color_table(3G), fill_color(3G), interior_style(3G), light_model(3G), polygon(3G), rectangle(3G), shade_mode(3G), vertex_format(3G), *Starbase Graphics Techniques*.

NAME

flush_buffer – output buffered primitives to display and return *without waiting* for display hardware to finish

SYNOPSIS

C Syntax:

```
void flush_buffer(fildes);
int fildes;
```

FORTRAN77 Syntax:

```
subroutine flush_buffer(fildes)
integer*4 fildes
```

Pascal Syntax:

```
procedure flush_buffer(fildes:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

Discussion

Flush_buffer outputs any previous graphics primitives that may still be in the device or driver buffers to the display device **fildes**. This command differs from *make_picture_current* in that it does not wait for the display device to finish its processing before returning. This means that *make_picture_current* will not return until all the pixels are on the screen and error checking is complete. Note that *flush_buffer* will update the picture just as fast as *make_picture_current*, it simply does not poll the hardware or sleep, waiting for completion, like *make_picture_current* does.

The use of *flush_buffer* instead of *make_picture_current* can lead to significant performance improvements on accelerated devices since the CPU can continue doing useful work while the graphics device is finishing its processing. This parallel processing capability greatly enhances performance. This command is especially useful for interactive graphics sessions at points where the operator must see the entire image in order to proceed.

Caution should be exercised not to overuse this command (or *make_picture_current*) since excessive flushing of system buffers will severely impact performance.

SEE ALSO

buffer_mode(3G), gclose(3G), make_picture_current(3G).

NAME

flush_matrices – flush matrix stack; reset viewing transformation matrix

SYNOPSIS

C Syntax:

```
void flush_matrices(fildes);
int fildes;
```

FORTRAN77 Syntax:

```
subroutine flush_matrices(fildes)
integer*4 fildes
```

Pascal Syntax:

```
procedure flush_matrices(fildes : integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

Discussion

All matrices in the matrix stack are removed and the current viewing transformation matrix is set equal to the current vdc-to-device transformation. This is the matrix that converts from virtual device coordinates to device coordinates (each device type requires its own unique matrix). Modelling and world coordinates are then equivalent to virtual device coordinates.

Flush_matrices is the only mechanism available for resetting the viewing transformation matrix back to the vdc-to-device transformation matrix once it has been altered by *view_camera*, *view_matrix*, *view_volume*, or *view_window*.

Subsequent calls to *view_port* do not cause a recalculation of the viewing transformation matrix until a viewing operation (*view_camera*, *view_volume*, or *view_window*) is called.

SEE ALSO

view_matrix(3G), view_port(3G).

NAME

gclose – close I/O path and release all resources assigned to specified graphics device

SYNOPSIS

C Syntax:

```
int gclose(fildes);
int fildes;
```

FORTRAN77 Syntax:

```
integer*4 function gclose(fildes)
integer*4 fildes
```

Pascal Syntax:

```
function gclose(fildes : integer): integer;
```

DESCRIPTION

Input Parameters

fildes

Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

Discussion

Gclose closes a graphic device opened with *gopen*. If successful, a value of 0 (zero) is returned. Otherwise, a value of −1 is returned and *gerr* is set to indicate the error.

Gclose attempts to flush the Starbase buffer (like *make_picture_current*), close the specified device and release all assigned resources regardless of errors detected.

SEE ALSO

gopen(3G), make_picture_current(3G).

NAME

gerr_control – control the handling of a graphics error.

SYNOPSIS

C Syntax

void gerr_procedure(proc);
void (*proc)();

void gerr_defaults();

void gerr_print_control(print_gerr)
int print_gerr;

char *gerr_message(errnumber);
int errnumber;

FORTRAN77 SYNTAX

subroutine gerr_procedure(proc)
integer*4 function proc

subroutine gerr_defaults()

subroutine gerr_print_control(print_gerr)
integer*4 print_gerr

subroutine gerr_message(errnumber, mesg)
character*(*) mesg
integer*4 errnumber

PASCAL SYNTAX

procedure gerr_procedure(procedure proc);

procedure gerr_defaults();

procedure gerr_print_control(print_gerr:integer);

procedure gerr_message(errnumber:integer;var mesg:string);

DESCRIPTION

INPUT PARAMETERS

proc is a pointer to the user supplied procedure to be called when an error is detected. If the pointer is null then the default procedure will be called.

print_gerr is an integer value which determines if errors and/or warnings or neither are to be printed.

errnumber is an error number

DISCUSSION

Gerr_control is used to specify a user-supplied procedure to be called when an error is detected. If a null pointer is passed, then the Starbase system procedure will be called. For systems which find it difficult (or impossible) to pass a null procedure pointer, the *gerr_defaults* procedure is provided which re-instates the default conditions listed below.

Gerr_message returns the text of an error message. The maximum length of a message is 80 characters. The control of error message printing is handled by the *gerr_print_control* procedure.

The values for *print_gerr* are:

NO_ERROR_PRINTING
PRINT_ERRORS
PRINT_WARNINGS

and may be or'ed together for desired combinations.

DEFAULTS

The Starbase system error procedure is called when an error is detected.
Print-gerr = PRINT_ERRORS (only errors are printed, warnings are not printed.)

SEE ALSO

inquire_gerror(3g).

NAME

gescape – input or output to device in a device-dependent manner

SYNOPSIS

C Syntax:

```
void gescape(fildes,op,arg1,arg2)
int fildes, op;
gescape_arg *arg1,*arg2;
```

FORTRAN77 Syntax:

```
subroutine gescape(fildes,op,arg1,arg2)
integer*4 fildes,op,
integer*4 arg1(64),arg2(64)
or
real arg1(64),arg2(64)
or
character arg1(255),arg2(255)
```

Pascal Syntax:

```
procedure gescape(fildes,op:integer;
var arg1,arg2:gescape_arg);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

op Operation to be performed on the device.

Input/Output Parameters

arg1 and **arg2** Pointers to argument lists.

Discussion

The device driver is called with the operation code and pointers to the two argument lists. The results of the operation are device-dependent and are explained in the Starbase Device Driver Library.

In C, *gescape_arg* is defined in <**starbase.c.h**> as:

```
typedef union {
    int   i[64];
    float f[64];
    char  c[255];
} gescape_arg;
```

In Pascal, *gescape_arg* is defined in <**starbase.p1.h**> as:

```
gescape_arg = record
        case integer of
            0: (i : array[1..64] of integer);
            1: (f : array[1..64] of real);
            2: (c : array[1..255] of char);
        end;
```

SEE ALSO

Starbase Device Drivers Library.

NAME

gopen – open I/O path to, create environment for, and initialize graphics device

SYNOPSIS

C Syntax:

```
int gopen (path,kind,driver,mode);
char *path,*driver;
int kind,mode;
```

FORTRAN77 Syntax:

```
integer*4 function gopen(path,kind,driver,mode)
character*(*) path,driver
integer*4 kind,mode
```

Pascal Syntax:

```
function gopen(path:string255;kind:integer;
        driver:string255;mode:integer):integer;
```

DESCRIPTION

Input Parameters

path Device file name for the device to be opened. This file is usually found in the /dev directory.

kind One of the constants:

INDEV	Device is to be used for input only.
OUTDEV	Device is to be used for output only.
OUTINDEV	Device is to be used for both input and output.

driver Character representation of the hardware device type. See *Starbase Device Drivers Library* for details.

mode Word containing flags used at open time to specify control. The following flags can be bitwise ORed together:

SPOOLED	Output is spooled to a file. Do not inquire from the device. If this bit is set, the *path* parameter should name a regular file or FIFO special file.
RESET_DEVICE	Device is completely initialized, including color map initialization and clearing the view surface.
INIT	Device is initialized in a device-dependent manner.
THREE_D	All transformations are three dimensional, including point, line, and polygon transformations as well as matrix concatenations.
MODEL_XFORM	As part of the opening, the device is prepared for shading of output primitives and/or removal of back-facing polygons. This means that output primitives must be transformed in two stages:

1. All output primitives are transformed by the top matrix on the matrix stack (if any matrices have been pushed onto the stack). This transforms the primitives from modelling coordinates to world coordinates. At this stage rendering calculations can be made.
2. Primitives are transformed from world coordinates to device coordinates by the current viewing and vdc-to-

device units transformations. If the device is not opened in this mode, all transformations can be concatenated and performed simultaneously.

INT_XFORM — Only Integer and common operations will be performed. All Floating point operations will cause an error.

FLOAT_XFORM — Only Floating point and common operations will be performed. All Integer operations will cause an error.

Discussion

This function returns a non-negative integer upon a successful device opening. This integer is called the *file descriptor* and is referred to by the name *fildes* in this manual.

The file descriptor remains open across *exec* system calls. See *fcntl*(2). If the *SPOOLED* bit is set, the file will be created or overwritten.

No process can have more than *_NFILE* file descriptors open simultaneously. *_NFILE* is an HP-UX system variable that defines the number of open files allowed per process at any given time. The value assigned to this variable is found in */usr/include/stdio.h*.

When a graphics device is opened, many defaults are set. They are described more fully in the file */usr/lib/starbase/defaults*.

The device file name *path* is created by the *mknod* command. For information about creating the device file, consult the *Starbase Device Drivers Library* manual in the chapter describing the device of interest.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode Floating point operations are not available for that fildes. Floating point operations are available by the default, or can be specified with FLOAT_XFORM mode. For a list of Integer operations, Floating point operations and common operations see the *starbase.3g* manual page.

RETURN VALUE

Upon successful completion, a non-negative integer called the file descriptor (*fildes*) is returned. Otherwise, a value of -1 is returned and *errno* is set to indicate the error.

EXAMPLES

The following example opens an HP 98720 display device for spooled output and performs device initialization:

```
fildes=gopen("/dev/crt",OUTDEV,"hp98720",INIT|THREE_D|MODEL_XFORM);
```

SEE ALSO

errno(2), gclose(3G), make_x11_gopen_string(3G), *Starbase Device Drivers Library Manual, Starbase Programming with X11.*

NAME

hatch_orientation – specify hatch line orientation

SYNOPSIS

C Syntax:

```
void hatch_orientation(fildes,vector_x,vector_y)
int fildes;
float vector_x,vector_y;
```

FORTRAN77 Syntax:

```
subroutine hatch_orientation(fildes,vector_x,vector_y)
integer*4 fildes
real vector_x,vector_y
```

Pascal Syntax:

```
procedure hatch_orientation(fildes:integer;vector_x,vector_y:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

vector_x,vector_y
A vector indicating the orientation of the hatch lines. For instance, (4.0,4.0) would specify hatching at a 45 degree angle.

Discussion

The hatch_orientation call allows the user to specify any desired orientation of hatch lines. The orientation vector is relative to the display surface; therefore, hatchlines drawn with an orientation of (1.0,1.0) will always be at an angle of 45 degrees. An orientation vector of (0.0,0.0) will return an error.

Defaults

vector_x,vector_y = (1.0,0.0).

SEE ALSO

hatch_type(3G), hatch_spacing(3G), interior_style(3G)

NAME

hatch_spacing – specify spacing between hatch lines

SYNOPSIS

C Syntax:

```
void hatch_spacing(fildes,spacing,mode)
int fildes,mode;
float spacing;

void inthatch_spacing(fildes,spacing,mode)
int fildes,spacing,mode;
```

FORTRAN77 Syntax:

```
subroutine hatch_spacing(fildes,spacing,mode)
integer*4 fildes,mode
real spacing

subroutine inthatch_spacing(fildes,spacing,mode)
integer*4 fildes,spacing,mode
```

Pascal Syntax:

```
procedure hatch_spacing(fildes:integer; spacing:real; mode:integer);

procedure inthatch_spacing(fildes,spacing,mode:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

spacing is the spacing between hatchlines in coordinates indicated by **mode.**

mode is one of: DC_UNITS, VDC_UNITS, MC_UNITS.

Discussion

Hatch_spacing allows the user to specify the distance between adjacent hatchlines, measured in the direction perpendicular to the hatchlines. **Mode** specifies what units **spacing** is in; DC_UNITS if **spacing** is in device coordinates; VDC_UNITS if **spacing** is in virtual device coordinates; and MC_UNITS if **spacing** is in modelling coordinates.

Defaults

spacing =1/32 of the default *vdc_extent* x-axis.

mode = VDC_UNITS.

SEE ALSO

hatch_orientation(3G), hatch_type(3G), interior_style(3G)

NAME

hatch_type – specify type of hatching to be done

SYNOPSIS

C Syntax:

```
void hatch_type(fildes,style)
int fildes,style;
```

FORTRAN77 Syntax:

```
subroutine hatch_type(fildes,style)
integer*4 fildes,style
```

Pascal Syntax:

```
procedure hatch_type(fildes,style:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

style One of the following: PARALLEL_HATCH, CROSSHATCH

Discussion

This call allows the user to specify whether hatching is to be done with parallel lines or with 90 degree crosshatching.

Defaults

style = PARALLEL_HATCH

SEE ALSO

hatch_orientation(3G), hatch_spacing(3G), interior_style(3G)

NAME

hidden_surface – enable/disable hidden surface removal

SYNOPSIS

C Syntax:

```
int hidden_surface(fildes,hsr_on,cull);
int fildes,hsr_on,cull;
```

FORTRAN77 Syntax:

```
integer*4 function hidden_surface(fildes,hsr_on,cull)
integer*4 fildes,hsr_on,cull
```

Pascal Syntax:

```
function hidden_surface(fildes,hsr_on,cull:integer):integer;
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

hsr_on Flag to enable/disable hidden surface processing. If **hsr_on** is TRUE (1), hidden surface processing is enabled. If FALSE (0), hidden surface processing is disabled.

cull Flag to enable/disable backfacing polygon culling. If **cull** is TRUE (1), culling is enabled; if FALSE (0), culling is disabled.

Discussion

Hidden_surface specifies whether or not hidden surface removal is to be performed. If **hsr_on** is FALSE (0), output primitives are drawn without regard to hidden surface processing. If **hsr_on** is TRUE (1), output primitives are drawn using z-buffer hidden surface removal.

Hidden_surface returns the number of times polygon information must be sent for a complete picture. If the function returns a number greater than one, the data must be sent *n* times and calls must be made to *zbuffer_switch* to render the entire area. This number is a function of the size of the display area to be used for hidden-surface-removal rendering and the size of the z-buffer. The portion of the display surface to be used for hidden-surface-removal rendering is defined by the viewport limits (see *view_port*), and the size of the z-buffer can be set in a device-dependent manner using *gescape*.

When *hidden_surface* is enabled on a device that has a z-buffer smaller than the entire display, the clip limits are set to the intersection of the z-buffer limits, the view port (as set by *view_port*), and the current clip limits (as set by *clip_indicator*). On devices which have a z-buffer the same size as the display, the clip limits are left as the current clip limits set by *clip_indicator*.

The z-buffer may be cleared with calls to *zbuffer_switch* or *clear_view_surface*. See these two routines and *clear_control* for details.

When hidden surface removal is enabled, functions that alter the z-buffer configuration (and hence, possibly the number of passes needed for hidden surface removal) are not allowed. This includes any functions that change vdcs (also changing the size of the view_port). Functions that generate an error if called with hidden surface enabled are *shade_mode*, *double_buffer*, *view_port*, *vdc_extent*, *mapping_mode*, *set_p1_p2*, and *vdc_justification*.

If **cull** is TRUE (1) and the graphics device has been opened in MODEL_XFORM mode (see *gopen*), polygons with normals facing away from the viewer are not rendered. The normals may be generated or supplied with each polygon (see *vertex_format*). If **cull** is FALSE (0), all polygons are rendered. Note that backface color as set by the **color** parameter of *backface_control*(3G) has no effect on backfacing polygons unless **cull** is FALSE.

In order for back culling to work properly with perspective views, the position of the viewer must be set with either *view_camera* or *viewpoint*. If the user is not using *view_camera*, but is doing his own perspective viewing matrix using *view_matrix3d*, the viewpoint function must be called with a POSITIONAL viewpoint to set the position of the viewer. See *viewpoint*(3G).

Recommendation

Enable front and back clipping (use *depth_indicator*) when depth cueing is enabled.

DEFAULTS

hsr_on = FALSE (0): draw output primitives without hidden surface removal
cull = FALSE (0): all polygons rendered

SEE ALSO

backface_control(3G), clear_control(3G), clear_view_surface(3G), gopen(3G), vertex_format(3G), view_camera(3G), view_port(3G), viewpoint(3G), zbuffer_switch(3G), *Starbase Device Drivers Library Manual*, *Starbase Graphics Techniques*.

NAME

highlight – specify highlighting color, style, and attributes

SYNOPSIS

C Syntax:

```
void highlight_color_index(fildes, index);
int fildes, index;

void highlight_color(fildes, red, green, blue);
int fildes;
float red, green, blue;

void highlight_type(fildes, style);
int fildes, style;

void highlight_attributes(fildes, attrs);
int fildes, attrs;

void highlight_on(fildes, flag);
int fildes, flag;
```

FORTRAN77 Syntax:

```
subroutine highlight_color_index(fildes, index)
integer*4 fildes, index

subroutine highlight_color(fildes, red, green, blue)
integer*4 fildes, index
real red, green, blue

subroutine highlight_type(fildes, style)
integer*4 fildes, style

subroutine highlight_attributes(fildes, attrs)
integer*4 fildes, attrs

subroutine highlight_on(fildes, flag)
integer*4 fildes, flag
```

Pascal Syntax:

```
procedure highlight_color_index(fildes,index:integer);

procedure highlight_color(fildes:integer; red,green,blue:real);

procedure highlight_type(fildes,style:integer;);

procedure highlight_attributes(fildes,attrs:integer;);

procedure highlight_on(fildes,flag:integer;);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

index is the color map entry that designates the color to use for drawing primitives affected by highlight.

red,green,blue are floating point color intensity values that designate the color to use for drawing primitives affected by highlight.

style may be SOLID, DASH, DOT, DASH_DOT, DASH_DOT_DOT, LONG_DASH, CENTER_DASH, and CENTER_DASH_DASH -- see *line_type* for more information on styles.

attrs is HL_COLOR, HL_STYLE, or (HL_COLOR | HL_STYLE).

HL_COLOR

- designates that color is to be used for highlighting text, markers, lines, filled-area edges, and hatched interiors

HL_STYLE

- designates that a line style is to be used for highlighting text, markers, lines, filled-area edges, and hatched interiors

flag is TRUE or FALSE.

TRUE

- turns highlight on
- postpones the effect of all attribute routines that are overridden by the *highlight_attributes'* **attrs** value (see discussion below)
- will NOT affect the rendering of filled-areas

FALSE

- turns highlight off
- restores attributes to what they would have been had highlight never been turned on
- enables all attribute routines to take effect immediately when they are called (see discussion below)

Discussion

Highlight_color_index allows the user to designate the color to be used when highlighting is on and HL_COLOR is in effect. This routine can NOT be called while highlighting is turned on; Starbase will generate a warning if it is called while highlighting is active.

Highlight_color is identical to *highlight_color_index* except that it allows the user to specify the highlight color in terms of floating point RGB values. This routine can NOT be called while highlighting is turned on; Starbase will generate a warning if it is called while highlighting is active.

Highlight_style allows the user to designate the line style to be used when highlighting is on and HL_STYLE is in effect. This routine can NOT be called while highlighting is turned on; Starbase will generate a warning if it is called while highlighting is active.

Highlight_attributes allows a user to designate how highlighting will be done. It is recommended that HL_STYLE be used only when color will not be effective -- for example, when highlighting must be done on a monochrome device. This routine can NOT be called while highlighting is turned on; Starbase will generate a warning if it is called while highlighting is active.

HL_COLOR has the following effects while highlighting is on:

- overrides *line_color(_index)*, *marker_color(_index)*, *text_color(_index)*, *perimeter_color(_index)*, and *bf_perimeter_color(_index)* color attribute routines using **index** or **rgb**
- forces the **edged** parameter in *interior_style* and *bf_interior_style* to TRUE, resulting in polygons and other filled primitives being edged in the highlight color. If HL_STYLE is active it is edged with the highlight style, otherwise it is edged with the current perimeter style.
- does not override *fill_color(_index)* or *bf_color(_index)* for filled-area color, but does override these for hatching color
- does not override vertex data such as rgb or normals per vertex (use highlight_type to highlight primitives with vertex data).

HL_STYLE has the following effects while highlighting is on:

- overrides the *line_type*, *perimeter_type*, and *bf_perimeter_type* style attribute routines using **style**
- applies a style of DOT to the strokes drawn for highlighted text, markers, and hatching patterns
- applies a device dependent repeat-length to highlighted text, markers, and hatching patterns
- forces the **edged** parameter in *interior_style* and *bf_interior_style* to TRUE, resulting in polygons and other filled primitives being edged with the highlight style. They will be edged in the highlight color if HL_COLOR is also active.
- if HL_COLOR is not active, the the **style** parameter in *interior_style* and *bf_interior_style* is forced to INT_HOLLOW, causing fill-area primitives to be outlined in the style specified and not filled, making the outline visible

Highlight_on allows the user to turn on and off highlight.

The following table summarizes the effect of *highlight_attributes*:

Primitive	HL_COLOR	HL_STYLE	HL_COLOR and HL_STYLE
Lines	color	style (1)	color and style (1)
Text	color	DOT (1)	color and DOT (1)
Markers	color	DOT (1)	color and DOT (1)
Edging	forced on	forced on	forced on
Polygon Edges	color	style (2)	color and style (2)
Interior style INT_SOLID	no change	HOLLOW	no change
Interior style INT_HATCH	color	HOLLOW	color and DOT (1)
Interior style INT_PATTERN	no change	HOLLOW	no change

(1) *line_repeat_length* affects the repeat length of *line_type*.
(2) *perimeter_repeat_length* (or *bf_perimeter_repeat_length*) affects the repeat length of polygon edges.

DEFAULTS

index	=	2	**style**	=	DOT
red	=	1.0	**attrs**	=	HL_COLOR
green	=	0.0	**flag**	=	FALSE
blue	=	0.0			

SEE ALSO

interior_style(3G), line_color(3G), line_type(3G), marker_color(3G), perimeter_color(3G), perimeter_type(3G), text_color(3G).

NAME

hit_mode – enable or disable hit detection

SYNOPSIS

C Syntax:

```
void set_hit_mode(fildes,hit_mode)
int fildes,hit_mode;

void inquire_hit(fildes,hit)
int fildes,*hit;
```

FORTRAN77 Syntax:

```
subroutine set_hit_mode(fildes,hit_mode)
integer*4 fildes,hit_mode

subroutine inquire_hit(fildes,hit)
integer*4 fildes,hit
```

Pascal Syntax:

```
procedure set_hit_mode(fildes,hit_mode:integer);

procedure inquire_hit(fildes:integer; var hit:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

hit_mode If TRUE (1), Starbase hit detection mode is enabled.
If FALSE (0), hit detection is disabled.

Output Parameters

hit TRUE (1) if a previous Starbase primitive generated a hit. Otherwise it is FALSE (0).

Discussion

Set_hit_mode and *inquire_hit* provide a mechanism for detecting hits on primitives. These are useful in application programs that implement their own display list.

Set_hit_mode is used to enable and disable the Starbase hit mode. When in hit mode, primitives are not displayed, but are tested for a hit. The status of this hit testing is saved for return by *inquire_hit* . If a hit has occurred since the last invocation of *inquire_hit*, **hit** will be TRUE (1). Calling *inquire_hit* resets the status of the hit test to FALSE (0).

A hit on a primitive occurs when a visible primitive intersects the pick aperture (see *set_pick_window*(3G) and *set_pick_depth*(3G)). Here is a list of pickable objects:

append_text
arc
block_move
block_write
draw2d
draw3d
ellipse
file_to_bitmap
file_to_intbitmap
intarc
intblock_move
intblock_write
intcircle
intdraw2d
intpolycircle
intpolygon2d
intpolyline2d
intpolymarker2d
intpolyrectangle
intrectangle
inttext2d
polygon2d
polygon3d
polyline2d
polyline3d
polymarker2d
polymarker3d
quadrilateral_mesh
rectangle
spline_curve2d
spline_curve3d
spline_surface
text2d
text3d
triangular_strip

Starbase Radiosity and Ray Tracing only:

spline_cone
spline_sphere
spline_torus

Note that the device coordinate (dc) primitives and *clear_view_surface* are not pickable, and never generate hits. *Partial_polygon*, *partial_ellipse*, *partial_arc*, *intpartial_polygon2d*, *intpartial_arc*, and *intpartial_circle* primitives are not directly pickable because they do not produce immediate output. However, they can contribute to the pickable output produced by a subsequent *polygon*, *rectangle*, *ellipse*, *arc*, *intpolygon2d*, *intrectangle*, *intcircle*, or *intarc* call. The same delayed reporting occurs for text primitives with the **more** flag set.

Hit_mode interacts with *hidden_surface*. If hit mode and hidden surface removal are both enabled, the z-buffer area is restricted to the area determined by *set_pick_window* and *set_pick_depth*. This may affect the number of *zbuffer_switch* calls required (See *hidden_surface*(3G)). Since a primitive in the background may be drawn before a primitive in the foreground obscures it, a hit may be recorded for a primitive that is not ultimately visible.

This can be prevented by drawing the complete image twice, and only calling *inquire_hit* during the second time through. The first time through sets the z-buffer values so that each primitive is compared with all other primitives on the second time through.

You may want to re-enable the Starbase hit mode after calling the *pick_from_segment* routine because *pick_from_segment* assumes the Starbase hit mode was turned on when it is called, and disables (turns off) the Starbase hit mode after executing, leaving the hit mode with possibly unexpected results.

ERRORS

1 Graphics device is not initialized for this operation.

DEFAULTS

Hit mode is disabled.

NOTE

This call can be used in both float (*gopen* with **FLOAT_XFORM**) and integer (*gopen* with **INT_XFORM**) mode.

SEE ALSO

pick_from_segment(3G), set_pick_window(3G), set_pick_depth(3G).

NAME

initiate_request – start request process without waiting for result

SYNOPSIS

C Syntax:

```
void initiate_request(fildes,class,ordinal,valid);
int fildes,class,ordinal,*valid;
```

FORTRAN77 Syntax:

```
subroutine initiate_request(fildes,class,ordinal,valid)
integer*4 fildes,class,ordinal,valid
```

Pascal Syntax:

```
procedure initiate_request(fildes,class,ordinal:integer;
var valid:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

class One of the following device classes: LOCATOR or CHOICE. The integer values assigned for these class designators are found in the Starbase **include** file used with your program.

ordinal Logical device number ranging from 1 to the number of devices in this class.

OUTPUT PARAMETER

valid If the device is not initialized or requests are not supported on a specific device, **valid** is set to FALSE (0). Otherwise it is set to TRUE (1). When Pascal is used, 0 and 1 are returned because TRUE and FALSE are reserved words in that programming language.

DESCRIPTION

If the device is supported, *initiate_request* posts a request against it. The request is completed when the device is triggered (in a device-dependent manner) or when the request is cancelled by receipt of a corresponding *request_choice* or *request_locator* that times out.

A request can be performed on an input device while it is enabled for event generation. The events-enabled mode is saved by the device and restored after the request has been executed.

Request data is not sent to the event queue, but is placed instead in a separate virtual register. *Inquire_request_status* can be used to determine whether any request data is present. Request data can be read by using *request_choice* or *request_locator* (*request_choice* and *request_locator* do not require an earlier call to *initiate_request* before they can be used).

Initiate_request provides a means for better utilizing devices that support asynchronous sampling during a pending request. Thus, a program can initiate a request then continue other operations without waiting for its completion.

SEE ALSO

inquire_request_status(3G), request_choice(3G), request_locator(3G).

NAME

inquire_cgm – inquire picture information from a cgm

SYNOPSIS

C Syntax:

```
void inquire_cgm(source,encoding,num_pictures)
char *source;
int *encoding,*num_pictures;

int inquire_cgm_names(source,start,count,response)
char *source;
int start, count;
char response[][256];
```

FORTRAN77 Syntax:

```
subroutine inquire_cgm(source,encoding,num_pictures)
character*(*) source
integer*4 encoding,num_pictures

integer*4 function inquire_cgm_names(source,start,count,response)
character*(*) source
integer*4 start, count
character*256 response(count)
```

Pascal Syntax:

```
procedure inquire_cgm(var source: string255;var encoding, num_pictures: integer);

function inquire_cgm_names(var source: string255;start, count: integer;
        var response: array[lo..hi:integer] of string255):integer;
```

DESCRIPTION

Input Parameters

source Pathname of the cgm file.

start Integer value indicating the picture number at which to start the picture names report.

count Integer value, equal to the number of picture names to report.

Output Parameters

encoding CGM_BINARY if the cgm is encoded in binary or TOP format. CGM_CHARACTER if the cgm encoded in character format. CGM_CLEAR_TEXT if the cgm is encoded in clear text format.

num_pictures The number of pictures stored in the metafile.

response Array holding the names of the pictures in the cgm.

Discussion

CGM is a graphical data base suitable for the storage and retrieval of picture information in a way that is compatible between devices of differing capabilities and design.

Inquire_cgm reads the cgm file specified by **source** and returns the encoding format and number of pictures in the metafile.

Inquire_cgm_names reads the cgm file specified by **source** and returns up to **count** metafile picture names in **response**, starting with picture number **start**. The function returns the actual number of metafile picture names returned. Picture number zero in the cgm is the metafile name. Picture number one in the cgm is the name of the first picture, picture number two is the name of the second picture, and so on up to the number of pictures in the metafile. Names will be truncated to 255 characters.

/usr/lib/libsbcgm.a must be linked to this program. See CGM in the *Starbase Graphics Techniques* manual.

SEE ALSO

cgm_to_starbase (3G), *Starbase Device Drivers Library Manual, Starbase Graphics Techniques.*

NAME

inquire_color_table – return current color table settings for specified graphic device

SYNOPSIS

C Syntax:

```
void inquire_color_table(fildes,start,count,response);
int fildes,start,count;
float response[][3];
```

FORTRAN77 Syntax:

```
subroutine inquire_color_table(fildes,start,count,response)
integer*4 fildes,start,count
real response(3,count)
```

Pascal Syntax:

```
procedure inquire_color_table(fildes,start,count:integer;
var response:array[lo..hi:integer]of rgb_color);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

start Integer index into the device color table at which to start the color table report. The first item in the color table is at index 0 (zero).

count Integer value, equal to the number of color table entries to report. Value range is one (1 entry) through the number of colors available on the device (256 for 256-color device). To determine the number of available color table entries on a device, use *inquire_sizes*.

Output Parameters

response Returns red, green, and blue values for each color table entry (values are in the range of zero (0.0) to one (1.0), where 1.0 is full intensity). In C terminology-

```
response[0][0] = color_table[start][0];
response[0][1] = color_table[start][1];
response[0][2] = color_table[start][2];
...
response[count][2] = color_table[start+count][2];
```

Discussion

Each file descriptor (**fildes**) opened as an output device has a color table associated with it. When this procedure is called, the color table is returned. If multiple file descriptors are open to the same device, the color table and the device's color map may not always be identical.

Some device drivers have a *gescape* (READ_COLOR_MAP) that reads the current color map into the color table. See the appropriate device driver description for more information.

For multi-bank graphics devices:

16 Planes	Each bank of 8 has the same color map.
24 Planes	If the mode parameter of *shade_mode* is set to CMAP_FULL and an inquire operation is performed on the color table, blue entries specify the intensities of bank 1, green applies to bank 2, and red to bank 3. Otherwise the device appears to be a standard 8-plane graphics device.

Multibank graphics devices that support video blending may allow separate color map definitions for different banks. See the *Starbase Device Drivers Library Manual* for details.

SEE ALSO

define_color_table(3G), rgb_to_index(3G), shade_mode(3G), *Starbase Device Drivers Library Manual.*

NAME

intinquire_current_position2d, inquire_current_position2d, inquire_current_position3d – return the current pen position.

SYNOPSIS

C Syntax:

```
void intinquire_current_position2d(fildes,x,y);
int fildes,*x,*y;

void inquire_current_position2d(fildes,x,y);
int fildes;
float *x,*y;

void inquire_current_position3d(fildes,x,y,z);
int fildes;
float *x,*y,*z;
```

FORTRAN77 Syntax:

```
subroutine intinquire_current_position2d(fildes,x,y);
integer*4 fildes,x,y;

subroutine inquire_current_position2d(fildes,x,y);
integer*4 fildes;
real x,y;

subroutine inquire_current_position3d(fildes,x,y,z);
integer*4 fildes;
real x,y,z;
```

Pascal Syntax:

```
procedure intinquire_current_position2d(fildes:integer;var x,y:integer);

procedure inquire_current_position2d(fildes:integer; var x,y:real);

procedure inquire_current_position3d(fildes:integer; var x,y,z:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

Output Parameters

x,y,z The current pen position in modelling coordinates.

Discussion

Intinquire_current_position2d, *inquire_current_position2d*, and *inquire_current_position3d* return the current modelling pen position. The current modelling pen position is defined to be the point where the last vector operation ended. The vector operations are *move2d*, *move3d*, *intmove2d*, *draw2d*, *draw3d*, *intdraw2d*, *polyline2d*, *polyline3d*, and *intpolyline2d*. All other commands leave the current position in a device-dependent location.

The current pen position for device coordinate operations is entirely different than that of modelling coordinates. Device coordinate current pen position is only consistent between device coordinate operations. After any modelling coordinate operation, a *dcmove* should be performed before any other device coordinate operations.

Integer operations are only available using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that file descriptor. Floating point operations are the default, or can be specified explicitly with FLOAT_XFORM mode. For a list of integer operations, floating point operations, and common operations, see the *starbase.3g* manual page.

SEE ALSO

Move(3G), draw(3G), polyline(3G), *Starbase Graphics Techniques.*

NAME

inquire_display_mode – return the current configuration for the display.

SYNOPSIS

C Syntax:

```
void inquire_display_mode(fildes,cmap_mode,dbuffer_mode,
planes,current_buffer);
int fildes,*cmap_mode,*dbuffer_mode,*planes,*current_buffer;
```

FORTRAN77 Syntax:

```
subroutine inquire_display_mode(fildes,cmap_mode,dbuffer_mode,
planes,current_buffer);
integer*4 fildes,cmap_mode,dbuffer_mode,planes,current_buffer;
```

Pascal Syntax:

```
procedure inquire_display_mode(fildes:integer;var cmap_mode,dbuffer_mode,
planes,current_buffer:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

Output Parameters

cmap_mode Specifies how the Starbase color map is setup. CMAP_NORMAL, CMAP_MONOTONIC, and CMAP_FULL are possible values.

dbuffer_mode Specifies whether double buffering is on or off and whether the buffer drawn to is the same as the one that is displayed. See **double_buffer(3G)** under *mode* for more specific information.

planes Specifies the number of planes in each buffer for input. The value may be 1, 2, 3, 4, 6, 8 or 12 (or 0 if **dbuffer_mode** is off).

current_buffer Least significant bit of this integer determines which buffer (odd(1) or even(0)) to use. See **dbuffer_switch(3G)** under *buffer* for more specific information.

Discussion

Inquire_display_mode can be used to determine how the information in the frame buffer will be displayed. It returns the current color map mode (as set by *shade_mode),* the current double buffering mode (as set by *double_buffer),* and which buffer is being displayed. This information is most useful when using an X11 window since some of these values may be different then the normal Starbase defaults.

SEE ALSO

double_buffer(3G), dbuffer_switch(3G), shade_mode(3G), *Starbase Graphics Techniques, Starbase Programming with X11.*

NAME

inquire_fb_configuration – inquire the memory configuration of the frame buffer

SYNOPSIS

C Syntax:

```
void inquire_fb_configuration(fildes,image_banks,image_planes,
                planes_per_bank,overlay_planes);
int fildes,*image_banks,*image_planes,*planes_per_bank,*overlay_planes;
```

FORTRAN77 Syntax:

```
subroutine inquire_fb_configuration(fildes,image_banks,
                image_planes,planes_per_bank,overlay_planes)
integer*4 fildes,image_banks,image_planes,
                planes_per_bank,overlay_planes
```

Pascal Syntax:

```
procedure inquire_fb_configuration (fildes:integer;
var image_banks,image_planes,planes_per_bank,overlay_planes:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

Output Parameters

image_banks Number of image banks supported by the device.

image_planes Total number of image planes supported by the device.

planes_per_bank Number of image planes in each bank

overlay_planes Total number of overlay planes supported by the device.

Discussion

The memory configuration of the frame buffer is returned in the output parameters. These values may be used in determining the range of viable parameter values for such procedures as *bank_switch*, *double_buffer*, and *shade_mode*.

Note that **planes_per_bank** times **image_banks** does not equal the number of image planes on devices that cannot display all banks simultaneously.

Devices with dedicated zbuffers may report zbuffer banks along with displayable banks in the **image_banks** parameter. This is to enable applications that wish to directly access the zbuffer a means to detect its presence. See the *Starbase Device Drivers Library Manual* for details.

SEE ALSO

bank_switch(3G), double_buffer(3G), shade_mode(3G), inquire_sizes (3G), *Starbase Device Drivers Library Manual*.

NAME

inquire_file – inquire the header information from a bitmapfile

SYNOPSIS

C Syntax:

```
void inquire_file(source,rev,device,xstart,ystart,
                  xlen,ylen, bm_mode,depth,pixel_align,num_banks,
                  cmap_mode,csize,back_index)
char *source;
int *rev;
char device[20];
int *xstart,*ystart,*xlen,*ylen;
int *bm_mode,*depth,*pixel_align,*num_banks;
int *cmap_mode,*csize,*back_index;
```

FORTRAN77 Syntax:

```
subroutine inquire_file(source,rev,device,
                  xstart,ystart,xlen,ylen,bm_mode,depth,
                  pixel_align,num_banks,cmap_mode,csize,back_index)
character*(*) source
integer*4 rev
character*(20) device
integer*4 xstart,ystart
integer*4 xlen,ylen
integer*4 bm_mode,depth,pixel_align,num_banks
integer*4 cmap_mode,csize,back_index
```

Pascal Syntax:

```
procedure inquire_file(var source:string255;var rev:integer;
                  var device:device_model;var xstart,ystart,
                  xlen,ylen,bm_mode,depth,pixel_align,num_banks,
                  cmap_mode,csize,back_index);
```

DESCRIPTION

Input Parameters

source Pathname of the file containing the raster information.

Output Parameters

rev Returns the revision number of the raster file format. The initial implementation is revision 1.

device Character string representing the type of bitmap from which the data was originally stored. For files generated by the Starbase *bitmap_to_file*(3G) procedure, this is the name of the device driver for the bitmap.

xstart,ystart Coordinates (in device coordinates) of the upper left corner of the source rectangle in the original bitmap. This is the corner of the rectangle actually stored, which may be different from the rectangle that was requested at the time of the file creation if clipping occurred.

xlen, ylen Length, in pixels, of the sides of the rectangle represented in the file.

bm_mode File format identifier.

- If the value of **bm_mode** is PIXEL_MAJOR (−1), the file is in pixel-major format.
- If the value of **bm_mode** is PLANE_MAJOR (−2), the format is plane-major.
- If **bm_mode** is non-negative, it is the plane number of a single plane, stored in the file in packed, plane-major format.

depth Total number of bits per pixel stored in the file. If the file format is **pixel_major** and **depth** is greater than eight, the data is stored in multiple banks, each having 8 bits/pixel. If the file format is **plane-major**, **depth** is equivalent to the number of planes stored in the file.

pixel_align Alignment of the pixel data in each bank or plane. If the value of **pixel_align** is **1**, every bit is a pixel. A value of eight means every byte is a pixel.

num_banks Number of banks of pixel data in the file. This parameter is relevant only for pixel-major file format.

cmap_mode Shade mode in effect when the bitmap data was stored. If the bitmap was stored with a normal color map, **cmap_mode** is set to CMAP_NORMAL (0). If the bitmap was stored with a monotonic color map, **cmap_mode is set to** CMAP_MONOTONIC (1). If the bitmap was stored with a full color map, **cmap_mode** is set to CMAP_FULL (4).

csize Number of color table entries stored in the file. A value of zero means that no color table was stored.

back_index Background color index in effect at the time the bitmap data was stored.

Failure causes include file not found and file does not contain raster data.

SEE ALSO

bitmap_to__file(3G), file_to__bitmap(3G), file_print(3G), shade_mode(3G), *Starbase Device Drivers Library Manual.*

NAME

inquire_gerror – return information on the most recent graphics error

SYNOPSIS

C Syntax:

```
void inquire_gerror (error,error_fildes);
int *error,*error_fildes;
```

FORTRAN77 Syntax:

```
subroutine inquire_gerror(error,error_fildes)
integer*4 error,error_fildes
```

Pascal Syntax:

```
procedure inquire_gerror(var error,error_fildes:integer);
```

DESCRIPTION

Output Parameters

error Graphics error number for the most recent error.

error_fildes File descriptor for the device specified in the library invocation where the error occurred.

NAME

inquire_id – return a unique device identifier and device-dependent information

SYNOPSIS

C Syntax:

```
void inquire_id(fildes,revision,model,type)
int fildes;
float *revision;
char model[20];
int *type;
```

FORTRAN77 Syntax:

```
subroutine inquire_id(fildes,revision,model,type)
integer*4 fildes,type
real revision
character*20 model
```

Pascal Syntax:

```
procedure inquire_id(fildes:integer;var revision:real;
var model:device_model;type:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when the I/O path to the output graphics device is opened.

Output Parameters

revision Device driver revision number with the initial release being 1.0.

model Device model number.

type Type of device: hard-copy (0) or soft-copy (1).

NAME

inquire_input_capabilities – inquire capabilities of physical input device specified by *fildes*

SYNOPSIS

C Syntax:

```
void inquire_input_capabilities(fildes,events_queue,
        sample_while_request,number_of_locators,number_of_valuators,
        number_of_choice_devices,number_of_string_devices);
int fildes,*events_queue,*sample_while_request;
int *number_of_locators,*number_of_valuators;
int *number_of_choice_devices,*number_of_string_devices;
```

FORTRAN77 Syntax:

```
subroutine inquire_input_capabilities(fildes,
        event_queue,sample_while_request,number_of_locators,
        number_of_valuators,number_of_choice_devices,
        number_of_string_devices)

integer*4 fildes,event_queue,sample_while_request
integer*4 number_of_locators,number_of_valuators
integer*4 number_of_choice_devices
integer*4 number_of_string_devices
```

Pascal Syntax:

```
procedure inquire_input_capabilities(fildes:integer;
var event_queue,sample_while_request,number_of_locators,
        number_of_valuators,number_of_choice_devices,
        number_of_string_devices:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

Output Parameters

events_queue Set to the defined value:
TRUE (1) if an event queue is supported,
FALSE otherwise.

sample_while_request
Set to the defined value:
TRUE (1) if the device can be sampled while a request is pending,
FALSE otherwise.

number_of_locators
Set to the number of locator devices supported.

number_of_valuators
Always set to 0: valuators are not supported at this time.

number_of_choice_devices
Set to the number of choice devices supported.

number_of_string_devices
Always set to 0: string devices are not supported at this time.

Discussion

Inquire_input_capabilities inquires the input capabilities of the physical device specified by **fildes**. If the device exists and is initialized, the information specified is returned. If the device does not exist or is not initialized, **gerror** is set and the output parameters are undefined.

NAME

inquire_request_status – inquire status of a request to an input device

SYNOPSIS

C Syntax:

```
void inquire_request_status(fildes,class,ordinal,ready);
int fildes,class,ordinal,*ready;
```

FORTRAN77 Syntax:

```
subroutine inquire_request_status(fildes,class,ordinal,ready)
integer*4 fildes,class,ordinal,ready
```

Pascal Syntax:

```
procedure inquire_request_status(fildes,class,
        ordinal:integer;var ready:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

class One of the defined values, LOCATOR, CHOICE, or ALL.

ordinal Logical device number ranging from 1 to number of devices in this class.

OUTPUT PARAMETER

ready Set to TRUE (1) if there is a request ready but not yet read.

Discussion

Inquire_request_status returns the status of a pending request for the device named by **fildes**, **class**, and **ordinal**. If **class** is set to ALL, **ordinal** is ignored and **ready** returns the logical OR of **ready** for each logical device associated with this **fildes**.

Ready returns TRUE (1) if there is a request ready but not yet read, and FALSE (0) if there is no request pending or the request pending is not ready.

This call can be used to determine (poll) whether a *request_choice* or *request_locator* needs to be issued, without halting the calling process to wait for an input.

SEE ALSO

initiate_request(3G), request_choice(3G), request_locator(3G).

NAME

inquire_sizes – return device physical limits, resolution, (p1,p2) and color map size

SYNOPSIS

C Syntax:

```
void inquire_sizes(fildes,physical_limits,resolution,
        p1,p2,cmap_size);
int fildes;
float physical_limits[2][3],resolution[3],p1[3],p2[3];
int *cmap_size;
```

FORTRAN77 Syntax:

```
subroutine inquire_sizes(fildes,physical_limits,
        resolution,p1,p2,cmap_size)
integer*4 fildes,cmap_size
real physical_limits(3,2),resolution(3),p1(3),p2(3)
```

Pascal Syntax:

```
procedure inquire_sizes(fildes:integer;
var physical_limits:three_d_limits;
var resolution:three_d_resolution; var p1,p2:three_d_point;
var cmap_size:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

Output Parameters

physical_limits

Device's accessible area in device coordinates (z=0 for 2D devices), except for the hpgl devices physical limits, which are the **p1** and **p2** limits.

resolution Number of millimeters per device coordinate unit.

p1,p2 Physical device limits in millimetres. **p1** and **p2** are the points (lower-left and upper-right respectively) that define the device viewport that the *vdc_extent* is mapped to, using the *vdc_justification* and *mapping_mode*. The Starbase command *set_p1_p2* can be used to redefine **p1** and **p2** (z=0 for 2-Dimensional devices).

cmap_size Number of entries in the device's color map.

Discussion

The following values are returned:

physical_limit [0][0]:	Lower-left: *x*
physical_limit [0][1]:	Lower-left: *y*
physical_limit [0][2]:	Lower-left: *z*
physical_limit [1][0]:	Upper-right: *x*
physical_limit [1][1]:	Upper-right: *y*
physical_limit [1][2]:	Upper-right: *z*

The resolution, p1, and p2 values are returned in (x,y,z) order.

Note: the direction and origin of the device coordinates reported in **physical_limits** is different for different devices. For example, for bit-mapped displays, Windows/9000 windows, and X Window System windows, the device coordinate (0,0) is located in the upper left-hand corner of the window, with the coordinates increasing as you move downward and right. For graphics terminals, the device coordinate (0,0) is located in the lower left-hand corner of the window

with the coordinates increasing upward and right. For some plotters, the device coordinates are (0,0) in the middle.

SEE ALSO

mapping_mode(3G), vdc_extent(3G), vdc_justification(3G), set_p1_p2(3G).

NAME

inquire_text_extent, intinquire_text_extent2d – return text-extent rectangle coordinates

SYNOPSIS

C Syntax:

```
void inquire_text_extent(fildes,string,xform,extent)
int fildes,xform;
char *string;
float extent[12];

void intinquire_text_extent2d(fildes,string,xform,extent)
int fildes,xform,extent[8];
char *string;
```

FORTRAN77 Syntax:

```
subroutine inquire_text_extent(fildes,string,xform,extent)
integer*4 fildes,xform
character*(*) string
real extent(12)

subroutine intinquire_text_extent2d(fildes,string,xform,extent)
integer*4 fildes,xform,extent(8)
character*(*) string
```

Pascal Syntax:

```
type

string255=string[255]
text_extent_array=array[1..12] of real;
inttext2d_extent_array=array[1..8] of integer;

procedure inquire_text_extent(fildes:integer; str:string255;
        xform:integer; var extent:text_extent_array);

procedure intinquire_text_extent2d(fildes:integer; str:string255;
        xform:integer; var extent:inttext2d_extent_array);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

string String of ASCII characters.

xform Enumerated value specifying the type of transformation to perform on the font coordinates:

VDC_TEXT=vdc
WORLD_COORDINATE_TEXT=world coordinate
TOS_TEXT=top of matrix stack

OUTPUT PARAMETER

extent Return array for the coordinates of the concatenation point and the text extent rectangle.

Discussion

The *inquire_text_extent* and *intinquire_text_extent2d* procedures execute the string and return the extent values. *Inquire_text_extent* returns:

extent [0]: Concatenation point: *x* coordinate
extent [1]: Concatenation point: *y* coordinate

extent [2]: Concatenation point: *z* coordinate
extent [3]: Lower-left corner: *x* coordinate
extent [4]: Lower-left corner: *y* coordinate
extent [5]: Lower-left corner: *z* coordinate
extent [6]: Upper-left corner: *x* coordinate
extent [7]: Upper-left corner: *y* coordinate
extent [8]: Upper-left corner: *z* coordinate
extent [9]: Upper-right corner: *x* coordinate
extent [10]: Upper-right corner: *y* coordinate
extent [11]: Upper-right corner: *z* coordinate

Intinquire_text_extent2d returns the following values:

extent [0]: Concatenation point: *x* coordinate
extent [1]: Concatenation point: *y* coordinate
extent [2]: Lower-left corner: *x* coordinate
extent [3]: Lower-left corner: *y* coordinate
extent [4]: Upper-left corner: *x* coordinate
extent [5]: Upper-left corner: *y* coordinate
extent [6]: Upper-right corner: *x* coordinate
extent [7]: Upper-right corner: *y* coordinate

For VDC_TEXT the returned coordinates are in virtual device coordinate values.

For WORLD_COORDINATE_TEXT the returned values are in world coordinate values.

For TOS_TEXT the returned values are defined by the top of the matrix stack.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

SEE ALSO

text(3G), text_precision(3G).

NAME

inquire_text_extent, intinquire_text_extent2d – return text-extent rectangle coordinates

SYNOPSIS

C Syntax:

```
void inquire_text_extent(fildes,string,xform,extent)
int fildes,xform;
char *string;
float extent[12];

void intinquire_text_extent2d(fildes,string,xform,extent)
int fildes,xform,extent[8];
char *string;
```

FORTRAN77 Syntax:

```
subroutine inquire_text_extent(fildes,string,xform,extent)
integer*4 fildes,xform
character*(*) string
real extent(12)

subroutine intinquire_text_extent2d(fildes,string,xform,extent)
integer*4 fildes,xform,extent(8)
character*(*) string
```

Pascal Syntax:

```
type

string255=string[255]
text_extent_array=array[1..12] of real;
inttext2d_extent_array=array[1..8] of integer;

procedure inquire_text_extent(fildes:integer; str:string255;
        xform:integer; var extent:text_extent_array);

procedure intinquire_text_extent2d(fildes:integer; str:string255;
        xform:integer; var extent:inttext2d_extent_array);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

string String of ASCII characters.

xform Enumerated value specifying the type of transformation to perform on the font coordinates:

VDC_TEXT=vdc
WORLD_COORDINATE_TEXT=world coordinate
TOS_TEXT=top of matrix stack

OUTPUT PARAMETER

extent Return array for the coordinates of the concatenation point and the text extent rectangle.

Discussion

The *inquire_text_extent* and *intinquire_text_extent2d* procedures execute the string and return the extent values. *Inquire_text_extent* returns:

extent [0]: Concatenation point: x coordinate
extent [1]: Concatenation point: y coordinate

extent [2]:	Concatenation point: *z* coordinate
extent [3]:	Lower-left corner: *x* coordinate
extent [4]:	Lower-left corner: *y* coordinate
extent [5]:	Lower-left corner: *z* coordinate
extent [6]:	Upper-left corner: *x* coordinate
extent [7]:	Upper-left corner: *y* coordinate
extent [8]:	Upper-left corner: *z* coordinate
extent [9]:	Upper-right corner: *x* coordinate
extent [10]:	Upper-right corner: *y* coordinate
extent [11]:	Upper-right corner: *z* coordinate

Intinquire_text_extent2d returns the following values:

extent [0]:	Concatenation point: *x* coordinate
extent [1]:	Concatenation point: *y* coordinate
extent [2]:	Lower-left corner: *x* coordinate
extent [3]:	Lower-left corner: *y* coordinate
extent [4]:	Upper-left corner: *x* coordinate
extent [5]:	Upper-left corner: *y* coordinate
extent [6]:	Upper-right corner: *x* coordinate
extent [7]:	Upper-right corner: *y* coordinate

For VDC_TEXT the returned coordinates are in virtual device coordinate values.

For WORLD_COORDINATE_TEXT the returned values are in world coordinate values.

For TOS_TEXT the returned values are defined by the top of the matrix stack.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

SEE ALSO

text(3G), text_precision(3G).

NAME

interior_style, bf_interior_style – select fill type and boundary visibility for subsequent filled area primitives and backfacing polygons

SYNOPSIS

C Syntax:

```
void interior_style(fildes,style,edged);
int fildes,style,edged;

void bf_interior_style(fildes,style,edged);
int fildes,style,edged;
```

FORTRAN77 Syntax:

```
subroutine interior_style(fildes,style,edged)
integer*4 fildes,style,edged

subroutine bf_interior_style(fildes,style,edged)
integer*4 fildes,style,edged
```

Pascal Syntax:

```
procedure interior_style(fildes,style,edged:integer);

procedure bf_interior_style(fildes,style,edged:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

style

INT_HATCH	for a hatched interior;
INT_HOLLOW	for a hollow interior;
INT_OUTLINE	for an outline using the normal fill color calculations;
INT_PATTERN	for a user-defined interior fill pattern;
INT_POINT	for a point plot using the normal fill color calculations;
INT_SOLID	for a solid filled interior;

edged Set to TRUE (1) if the boundary is to be drawn.
Set to FALSE (0) for no drawn boundary.
The edge is drawn with the current perimeter attributes or, for backfacing polygons, the current backfacing perimeter color.

Discussion

Hollow style areas are not filled.

Solid style areas are filled with the current *fill_color* or *bf_fill_color*.

(Currently, only the 98731 driver supports these two features)

Outline style areas are drawn by outlining the fill area's border using all of the normal fill color calculations, including light sourcing and shading. Lines are generated at the intersections of the fill area with any of the current clip limits. This interior style can be used to generate light sourced, shaded wireframe images.

Point style areas are drawn as a point plot of the fill area's vertices using all of the normal fill color calculations, including light sourcing and shading. Points are generated at the intersections of the fill area with any of the current clip limits.

(If edging is enabled, polygon edges will be drawn over outline and point style filling.)

Polygons

Hatched polygons are filled in the current fill color with the hatch attributes defined by *hatch_type*, *hatch_spacing*, and *hatch_orientation*. When hatching is enabled, polygons are replaced by a set of vectors, either parallel or crossing. For this reason, several features normally associated with polygons are *not supported* when hatching is selected. These include hidden surface removal, lighting models, shading, depth cueing, backface culling, and backface control.

Setting **edged** equal to TRUE(1) displays a boundary of the current *perimeter_color*, and will obscure INT_OUTLINE or INT_POINT.

User-defined patterns are specified through the *pattern_define* call.

The *perimeter_type* and *perimeter_repeat_length* routines set the type and repeat length for frontfacing polygon edges.

Backfacing polygons

Backfacing polygons use the specified interior **style** when *bf_control* has been called with the **attr** parameter set to TRUE(1).

INT_PATTERN and INT_HATCH interior styles are not supported for backfacing polygons.

Setting **edged** equal to TRUE(1) displays a boundary of the current *bf_perimeter_color*, and will obscure INT_OUTLINE or INT_POINT.

The *bf_perimeter_type* and *bf_perimeter_repeat_length* routines set the type and repeat length for backfacing polygon edges.

DEFAULTS

INT_SOLID, not edged.

SEE ALSO

arc(3G). bf_control(3G), circle(3G), ellipse(3G), fill_color(3G), hatch_orientation(3G), hatch_spacing(3G), hatch_type(3G), pattern_define(3G), perimeter_color(3G), perimeter_repeat_length(3G), perimeter_type(3G), polygon(3G), rectangle(3G), *Starbase Graphics Techniques.*

NAME

intra_character_space – specify spacing between character cells

SYNOPSIS

C Syntax:

```
void intra_character_space(fildes,space);
int fildes;
float space;
```

FORTRAN77 Syntax:

```
subroutine intra_character_space(fildes,space)
integer*4 fildes
real space
```

Pascal Syntax:

```
procedure intra_character_space(fildes:integer;space:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

space Fraction of height to be added to the normal space between adjacent characters.

Discussion

Intra_character_space specifies the inter-character space between adjacent character cells as a fraction of the character cell height defined by *character_height*. For example, if the current character height is 0.3 and *character_height* is called with **space** set to 0.5, the space between adjacent character cells is

$$(0.3 * 0.5) = 0.15$$

in virtual device coordinate units. Inter-character spacing is added along the character path.

DEFAULTS

space = 0.0: No space between character cells.

SEE ALSO

character_height(3G).

NAME

bezier_knots, default_knots, u_knot_vector, v_knot_vector – define knot vector(s) for drawing subsequent space curves or surfaces

SYNOPSIS

C Syntax:

```
void bezier_knots (fildes);
int fildes;

void default_knots (fildes);
int fildes;

void u_knot_vector(fildes, knot_vector, numpts);
int fildes, numpts;
float knot_vector[];

void v_knot_vector(fildes, knot_vector, numpts);
int fildes, numpts;
float knot_vector[];
```

FORTRAN77 Syntax:

```
subroutine bezier_knots(fildes)
integer*4 fildes

subroutine default_knots(fildes)
integer*4 fildes

subroutine u_knot_vector(fildes,knot_vector,numpts)
integer*4 fildes,numpts
real knot_vector()

subroutine v_knot_vector(fildes,knot_vector,numpts)
integer*4 fildes,numpts
real knot_vector()
```

Pascal Syntax:

```
procedure bezier_knots(fildes:integer);

procedure default_knots(fildes:integer);

procedure u_knot_vector(fildes:integer;var knot_vector:array[lo..hi:integer] of real;
        numpts:integer);

procedure v_knot_vector(fildes:integer;var knot_vector:array[lo..hi:integer] of real;
        numpts:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

knot_vector Vector defining the knots to be used in the processing of spline curves and surfaces.

numpts Number of entries in the knot vector.

Discussion

U_knot_vector and *v_knot_vector* specify the knot vectors to be used with subsequent *trimming_curve, spline_curve2d, spline_curve3d* and *spline_surface* procedures. *Default_knots* specifies that uniform (1.0 spacing), non-multiple knots are to be used. The first interval used in rendering the spline will be 0 to 1. This is a faster method of interpolating, because it is not necessary to recompute basis matrices for each knot sequence. These knot vectors are used in calculating basis matrices used for interpolating spline curves and surfaces. *bezier_knots* specifies that a knot vector with knots of multiplicity equal to the order of the curve or surface are at the beginning and end of the knot sequence and intermediate knots have multiplicity of (order − 1). The use of *bezier_knots* will, of course, render Bezier curves and surfaces. Using *default_knots* or *bezier_knots* is a faster method of interpolating, because it is not necessary to recompute basis matrices for each knot sequence.

For trimming curves and spline curves, only the *u* knot vector is used. Knot vector routines can also handle non-uniform splines, so knot vectors are not restricted to integer values. They can also start at values other than 0.0.

DEFAULTS

Uniform (1.0 spacing) non-multiple knots.

SEE ALSO

spline (3G), *Starbase Graphics Techniques.*

NAME

light_ambient – define ambient light color

SYNOPSIS

C Syntax:

```
void light_ambient(fildes,red,green,blue);
int fildes;
float red,green,blue;
```

FORTRAN77 Syntax:

```
subroutine light_ambient(fildes,red,green,blue)
integer*4 fildes
real red,green,blue
```

Pascal Syntax:

```
procedure light_ambient(fildes:integer;red,green,blue:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

red, green, blue Values (in the range of 0.0 to 1.0) that define the ambient light color.

Discussion

Light_ambient specifies the color of the ambient light that is to be used when shading filled polygons (*light_switch* is used to enable or disable ambient lighting).

DEFAULTS

White ambient light (r=1.0, g=1.0, b=1.0)

SEE ALSO

light_model(3G), light_source(3G), light_switch(3G), shade_mode(3G), surface_model(3G), vertex_format(3G), *Starbase Graphics Techniques.*

NAME

light_attenuation – define attenuation constants for POSITIONAL light sources

SYNOPSIS

C Syntax:

```
void light_attenuation(fildes,index,quadratic,c1,c2,c3);
int fildes,index,quadratic;
float c1,c2,c3;
```

FORTRAN77 Syntax:

```
subroutine light_attenuation(fildes,index,quadratic,c1,c2,c3)
integer*4 fildes,index,quadratic
real c1,c2,c3
```

Pascal Syntax:

```
procedure light_attenuation(fildes,index,quadratic:integer;c1,c2,c3:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

index Number of the POSITIONAL light source to be modified.

quadratic If TRUE (1), the attenuation constants specified by the *c1,c2,c3* parameters are substituted into the quadratic equation used to calculate light attenuation. If FALSE (0), the attenuation constants are not used.

c1 Constant term in the light attenuation quadratic equation.

c2 Linear term in the light attenuation quadratic equation.

c3 Quadratic term in the light attenuation quadratic equation.

Discussion

If the **quadratic** parameter is TRUE , the *light_attenuation* subroutine specifies the parameters used to determine the intensity of shaded polygonal objects, based on the distance between the light source and object. These parameters are the attenuation constants in the following quadratic equation:

$$La = \frac{1}{(c1 + c2 \mid Lp - Op \mid + c3 \mid Lp - Op \mid^2)}$$

where Lp is the light source position, and Op is the object position. The light attenuation factor La is then multiplied by the diffuse and specular terms of the lighting equation. The value of La is clamped at 1.0 to avoid increasing the total intensity of the image as a result of light attenuation.

If the **quadratic** parameter is FALSE then the attenuation constants specified by this routine are ignored. Instead, values for the quadratic equation are determined from parameters to the *light_model* routine. Refer to the *light_model (3G)* manual page for a description of these parameters.

The *shade_mode* routine must be called to turn shading on before *light_attenuation* is actually enabled.

DEFAULTS

The **quadratic** parameter is FALSE so the light attenuation constants are determined by the **atten** parameter to *light_model*.

SEE ALSO

light_model(3G), light_source(3G), shade_mode(3G), vertex_format(3G), *Starbase Graphics Techniques*.

NAME

light_model – modify aspects of POSITIONAL light sources

SYNOPSIS

C Syntax:

```
void light_model(fildes,index,attr,spot,atten,ang,x,y,z);
int fildes,index,attr,spot;
float atten,ang,x,y,z;
```

FORTRAN77 Syntax:

```
subroutine light_model(fildes,index,attr,spot,atten,ang,x,y,z)
integer*4 fildes,index,attr,spot
real atten,ang,x,y,z
```

Pascal Syntax:

```
procedure light_model(fildes,index,attr,spot:integer;atten,ang,x,y,z:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

index Number of the POSITIONAL light source to be modified; the total number of light sources is device dependent.

attr Sets the attributes of the POSITIONAL light source. Can be any of ATTEN_LIGHT, SPOT_LIGHT, or CONE_LIGHT ORed together.

spot Specifies the spotlight power of the POSITIONAL light source in the range 0 to 16383 if SPOT_LIGHT is set in **attr**.

atten Specifies the attenuation factor if ATTEN_LIGHT is set in **attr**. Must be greater than 0.0.

ang Specifies the angle of the cone in degrees, where the light is cut off if CONE_LIGHT is set in *attr*. Must be greater than 0.0.

x,y,z Specifies the direction the POSITIONAL light source is pointing. Note that this vector does not need to be normalized to unit length.

Discussion

Light_model modifies certain aspects of POSITIONAL light sources as defined by *light_source*. If the ATTEN_LIGHT bit is set, the POSITIONAL light source intensity falls off as **atten** divided by **R**, where **R** is the distance from the light source to the vertex. **Atten** can be any positive floating point value, including values greater than 1.0. Vertices at a distance of **atten** from the light source or closer are fully illuminated while more distant vertices are more dimly illuminated, according to the factor given above.

When the SPOT_LIGHT bit is set, the intensity of the light falls off as the cosine of **A** raised to the **spot** power, where **A** is the angle between the direction the light source is pointing and the vector from the vertex to the light source. The **spot** parameter specifies how focused the spot light is: the higher the number, the more focused the light.

The CONE_LIGHT function is comparable to placing an opaque circular lamp shade around the light source. The light abruptly stops when the angle between the direction the light source is pointing and the vector from the vertex to the light source is greater than **ang**.

Any of these functions can be defined separately or combined together to form a realistic light source.

The number of light sources is device dependent. Currently eight (8) light sources are supported on an HP 98721 device, and fifteen (15) light sources are supported on all other 3-D accelerated devices.

DEFAULTS

Simple light source as specified by *light_source* with no modifications.

SEE ALSO

light_ambient(3G), light_source(3G), light_switch(3G), shade_mode(3G), vertex_format(3G), *Starbase Graphics Techniques.*

NAME

light_source – define light source positions and colors

SYNOPSIS

C Syntax:

```
void light_source(fildes,index,type,red,green,blue,x,y,z);
int fildes,index,type;
float red,green,blue,x,y,z;
```

FORTRAN77 Syntax:

```
subroutine light_source(fildes,index,type,red,green,blue,x,y,z)
integer*4 fildes,index,type
real red,green,blue,x,y,z
```

Pascal Syntax:

```
procedure light_source(fildes,index,type:integer;red,green,blue,x,y,z:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

index Selects the light source to be modified; the total number of light sources is device dependent.

type Selects the light source type from DIRECTIONAL, POSITIONAL

red,green,blue Values (in the range 0.0 to 1.0) that define the color of the light source.

x,y,z **x,y,z** coordinate position of the light source or a vector pointing to the light source. Note that if this is a vector, it does not need to be normalized to unit length.

Discussion

Light_source specifies the location and color of different light sources which are then used in shading of filled polygons. Up to eight (8) different light sources can be specified on a HP 98721 device, and up to fifteen (15) on all other 3-D accelerated devices. These numbers do not include the ambient light source which is always index 0 and cannot be modified to a DIRECTIONAL or POSITIONAL type.

The **type** parameter can be one of the following:

DIRECTIONAL Point source at infinity. The **x,y,z** parameters specify a vector anchored at the origin and directed toward the light source.

POSITIONAL Point source close to viewing area. The **x,y,z** parameters specify the position of the light source in world coordinates.

Additional aspects of POSITIONAL light sources can be set with *light_model* .

DEFAULTS

DIRECTIONAL directed at x=0.0, y=0.0, z=−1.0 with red=1.0, green=1.0, blue=1.0

SEE ALSO

light_ambient(3G), light_model(3G), light_switch(3G), shade_mode(3G), surface_model(3G), vertex_format(3G), *Starbase Graphics Techniques.*

NAME

light_switch – enable/disable light sources

SYNOPSIS

C Syntax:

```
void light_switch(fildes,on);
int fildes,on;
```

FORTRAN77 Syntax:

```
subroutine light_switch(fildes,on)
integer*4 fildes,on
```

Pascal Syntax:

```
procedure light_switch(fildes,on:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

on Mask specifying which light sources are on (value range: 0 through 511).

Discussion

light_switch allows the point light sources and the ambient light source to be turned on and off independently. The bits in the **on** mask determine the state of each light source. The least significant bit (bit 0) specifies the ambient light, while bits 1 through 15 determine the state of point light sources 1 through 15 respectively. If a bit is set (1) the corresponding light source is on; if a bit is clear (0) the light source is off.

The number of light sources is device dependent. Currently eight (8) light sources are supported on an HP 98721 device, and fifteen (15) light sources are supported on all other 3-D accelerated devices.

DEFAULTS

on = 1: ambient light source is on if shading active.

SEE ALSO

light_ambient(3G), light_source(3G), shade_mode(3G), surface_model(3G), vertex_format(3G), *Starbase Graphics Techniques.*

NAME

line_color – select color index or color value for subsequent line primitives.

SYNOPSIS

C Syntax:

```
void line_color_index(fildes,index)
int fildes,index;

void line_color(fildes,red,green,blue)
int fildes;
float red,green,blue;
```

FORTRAN77 Syntax:

```
subroutine line_color_index(fildes,index)
integer*4 fildes,index;

subroutine line_color(fildes,red,green,blue)
integer*4 fildes;
real red,green,blue;
```

Pascal Syntax:

```
procedure line_color_index(fildes,index:integer);

procedure line_color(fildes:integer;red,green,blue:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when the I/O path to the output graphic device is opened.

index Integer index into the device color table. The line color is the red, green, and blue values specified at this table location. The size of the color table may be obtained using the *inquire_sizes* procedure.

red, green, blue
are the color values (in the range of 0.0 to 1.0) to be used for subsequent line primitives. Zero (0.0) indicates **no** color and one (1.0) indicates **full** color. Monochrome devices use the equation

$$I=0.30(red)+0.59(green)+0.11(blue)$$

to determine the intensity.

Discussion

To acquire the current color table definition, use the *inquire_color_table* command. To change the current color table definition, use the *define_color_table* procedure.

The *draw* and *polyline* primitives are affected by this procedure.

When the line color is set by using red, green, and blue parameters, determination of the actual output dependends on the state of the mode parameter set with *shade_mode*:

CMAP_NORMAL — This is the default mode if *shade_mode* has not been called. The color table is searched for an index that points to the color in RGB space that most closely matches the one specified. Specification by index is faster than color map searches.

CMAP_MONOTONIC — Color values are converted to intensity by using the equation:

$$I = 0.30(\text{red}) + 0.59(\text{green}) + 0.11(\text{blue}).$$

This intensity is then mapped to an index using the minimum and maximum defined by *shade_range*.

CMAP_FULL — Color values are mapped directly to a index with the assumption that the color map is set up to a predefined full color state.

If the color map is changed after this procedure is called, the line color used may not be the original color desired.

Color can also be specified in the polyline vertex list. See *vertex_format*(3G) for details.

If the index is out of range, a warning is generated and a mod function is performed.

DEFAULTS

The default line color is the color defined in the current color map at index 1.

SEE ALSO

define_color_table(3G), draw(3G), inquire_color_table(3G), marker_color(3G), perimeter_color(3G), polyline(3G), shade_mode(3G), text_color(3G).

NAME

line_endpoint – set a line endpoint type and corners for lines with width.

SYNOPSIS

C Syntax:

```
void line_endpoint(fildes,endpoint);
int fildes,endpoint;
```

FORTRAN77 Syntax:

```
subroutine line_endpoint(fildes,endpoint)
integer*4 fildes,endpoint
```

Pascal Syntax:

```
procedure line_endpoint(fildes,endpoint:integer);
```

DESCRIPTION

Input Parameters

fildes integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

endpoint an enumerated value specifying the type of endpoint for a line wider than the default line width:

SQUARE = endpoints are squared-off perpendicular to the primary line.

ROUNDED = endpoints are circular.

Discussion

Line_endpoint specifies the type of endpoint and corners that will be used for a *intpolyline2d*, or *intdraw2d* that has a specified width.

NOTE

Rounded end points with anisotropic mapping (x units do not equal y units) may produce unexpected, device-dependent results.

DEFAULT

Endpoint is system dependent.

SEE ALSO

draw(3G), polyline(3G), line_width(3G), vdc_extent(3G), *Starbase Graphics Techniques*.

NAME

line_repeat_length, intline_repeat_length – specify line pattern length for line primitives

SYNOPSIS

C Syntax:

```
void line_repeat_length(fildes,length);
int fildes;
float length;

void intline_repeat_length(fildes,length);
int fildes,length;
```

FORTRAN77 Syntax:

```
subroutine line_repeat_length(fildes,length)
integer*4 fildes
real length

subroutine intline_repeat_length(fildes,length)
integer*4 fildes,length
```

Pascal Syntax:

```
procedure line_repeat_length(fildes:integer;length:real);

procedure intline_repeat_length(fildes,length:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

length Repeat length of the line-type pattern measured in virtual device units along the x-direction.

Discussion

This procedure sets the repeat length for *polyline* and *draw* output primitives.

There are some device-dependent limitations for setting repeat length. Some raster devices round to the nearest multiple of 16 pixels.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

DEFAULT

Repeat **length** is 1/32 of the default *vdc_extent* x-axis.

SEE ALSO

line_type(3G), perimeter_repeat_length(3G).

NAME

line_type – select line type for all subsequent line primitives

SYNOPSIS

C Syntax:

```
void line_type(fildes,style);
int fildes, style;
```

FORTRAN77 Syntax:

```
subroutine line_type(fildes,style)
integer*4 fildes, style
```

Pascal Syntax:

```
procedure line_type(fildes,style:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

style One of the following:

SOLID
DASH
DOT
DASH_DOT
DASH_DOT_DOT
LONG_DASH
CENTER_DASH
CENTER_DASH_DASH

Discussion

Predefined line types are described below:

Index	Name	Approximate Pattern
0	SOLID	Solid
1	DASH	.25 .50 .25
2	DOT	4-8 dots per repeat length
3	DASH_DOT	.4 .1 dot .1 .35
4	DASH_DOT_DOT	.35 .1 dot .1 dot .1 .35
5	LONG_DASH	.375 .25 .375
6	CENTER_DASH	.35 .1 .1 .1 .35
7	CENTER_DASH_DASH	.25 .1 .1 .1 .1 .1 .25

Negative **style** values are device-dependent. For example, -1 is a point plot on a terminal.

Continuity between separate, but graphically connected, polyline or line segments is not guaranteed.

The pattern is restarted each time this procedure is executed.

Wide lines do not support line types.

Some devices do not support this procedure.

DEFAULTS

Solid Lines.

SEE ALSO

line_repeat_length(3G), perimeter_type(3G), *Starbase Device Drivers Library*.

NAME

intline_width, line_width – set line width.

SYNOPSIS

C Syntax:

```
void intline_width(fildes,width,mode);
int fildes,width,mode;

void line_width(fildes,width,mode);
int fildes,mode;
float width;
```

FORTRAN77 Syntax:

```
subroutine intline_width(fildes,width,mode)
integer*4 fildes,width,mode

subroutine line_width(fildes,width,mode)
integer*4 fildes,mode
real width
```

Pascal Syntax:

```
procedure intline_width(fildes,width,mode:integer);

procedure line_width(fildes:integer;width:real;mode:integer);
```

DESCRIPTION

Input Parameters

fildes integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

width is width of a line in units specified by mode.

mode a value specifying the units of width:

MC_UNITS = modelling coordinates
VDC_UNITS = virtual device coordinates

Discussion

Intline_width and *line_width* set the line width for subsequent *polyline2d*, *intpolyline2d*, *intdraw2d*, and *draw2d* calls.

If **mode** is **MC_UNITS**, *width* should be given in modelling coordinate units.

If **mode** is **VDC_UNITS**, *width* should be given in virtual device coordinate units.

Line width is measured by aligning a wide line with its ideal default-width defining line such that the distance between the defining line and either edge is half the line width.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that file descriptor. Floating point operations are the default, or can be specified explicitly with FLOAT_XFORM mode. Device cannot be *gopen*ed with the THREE_D **mode** for *line_width* to work; it is intended for 2D usage only. For a list of integer operations, floating point operations, and common operations, see the *starbase(3G)* manual page.

Note: *Line_type* does not apply to wide lines (width greater than zero). It is ignored. When the mapping of the VDC space to device is anisotropic because of a *vdc_extent* call, the rendered width of a line will change with the direction of the line segment.

DEFAULTS

width = 0.0; 1 pixel width.

SEE ALSO

draw(3G), line_endpoint(3G), polyline(3G), *Starbase Graphics Techniques.*

NAME

make_X11_gopen_string - create a path string associated with an existing X window

SYNOPSIS

C Syntax:

```
#include <X11/Xlib.h>

char *make_X11_gopen_string(display, drawable);
Display *display;
Drawable drawable;
```

FORTRAN77 SYNTAX

```
subroutine make_X11_gopen_string(display, drawable, dev_string)
    integer*4 display
    integer*4 drawable
    character*(*) dev_string
```

DESCRIPTION

Input Parameters

display The display where the drawable exists.

drawable The drawable that is to be gopened.

Output Parameters

dev_string The string, associated with an existing X11 window, that can be passed as the path parameter to *gopen*(3G).

Returned Value

If successful, *make_X11_gopen_string* returns the address of a string. If unsuccessful, *make_X11_gopen_string* returns NULL. For FORTRAN77 programs, the memory for the display string must be provided as a third parameter.

Discussion

This routine provides support for associating an existing X11 Window with a string that can be passed as the path parameter to *gopen*(3G).

The *display* argument specifies the X11 display that the drawable was created on. The *drawable* argument specifies the id of the Window that *gopen* should access. The memory for the string is obtained, by the routine, using *malloc*(3C), *malloc*(3X), and can be released using *free*(3C). This is not necessary for programs using the FORTRAN77 entry point.

Note that no Pascal syntax is shown above. This is because there are Xlib bindings for C and FORTRAN77 only, so a Pascal program would need to use the C bindings in order to call this routine. Refer to the appendixes of *Programming With Xlib* for more information.

SEE ALSO

gopen(3G), malloc(3C), malloc(3X), *Starbase Programming with X11*.

NAME

make_picture_current – output buffered primitives to display and wait for display hardware to finish

SYNOPSIS

C Syntax:

```
void make_picture_current(fildes);
int fildes;
```

FORTRAN77 Syntax:

```
subroutine make_picture_current(fildes)
integer*4 fildes
```

Pascal Syntax:

```
procedure make_picture_current(fildes:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

Discussion

Make_picture_current outputs any previous primitives that may still be in the device or driver buffers to the display device **fildes**. Next, the command waits for the display hardware to finish its processing in order to guarantee that when the subroutine returns, all pixels will be displayed.

It is commonly used:

- in interactive graphics sessions at points where the operator must see the entire image in order to proceed,
- or in hard-copy graphics at the end of the picture generation process such as when transferring an image from the raster buffer to the printed page in a buffered raster printer.

Gclose also performs a *make_picture_current* operation before closing the device.

Note that *make_picture_current* will degrade performance, especially on accelerated devices, since the CPU will wait for the graphics device to finish all of its processing. Therefore this procedure should be used with caution. For better performance, use *flush_buffer* which causes all output buffers to be flushed, but does not wait for the device to finish before returning. This allows more parallel processing with accelerated devices, while still bringing the picture up to date just as fast as *make_picture_current*.

SEE ALSO

buffer_mode(3G), flush_buffer(3G), gclose(3G).

NAME

mapping_mode – define vdc extent mapping to viewport as isotropic or anisotropic

SYNOPSIS

C Syntax:

```
void mapping_mode(fildes,distort);
int fildes,distort;
```

FORTRAN77 Syntax:

```
subroutine mapping_mode(fildes,distort)
integer*4 fildes,distort
```

Pascal Syntax:

```
procedure mapping_mode(fildes,distort:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

distort If TRUE (1), distorted mapping from virtual device coordinates to device coordinates is performed.

If FALSE (0), the mapping is not distorted.

Discussion

When *mapping mode* is distorted (anisotropic), the viewport occupies the full physical region of interest (set by *set_p1_p2*), with no space left unused in the region. This is referred to as a 2-point scaling or "rubbering" and does not preserve aspect ratio. The numbers corresponding to *vdc_extent* span the entire physical region of interest.

When *mapping mode* is not distorted (isotropic), the viewport is calculated as the largest rectangle within the physical region of interest that preserves the aspect ratio of the *vdc_extent*. Unless this matches the aspect ratio of the physical region of interest, some "white space" will be left unused outside the physical region of interest. By default, this white space is distributed equally on the two sides of the viewport unless modified by the *vdc_justification* procedure.

This procedure updates the current vdc-to-device units transformation matrix. If no matrices have been placed on the matrix stack and no viewing transformations have been defined, the new vdc matrix becomes the current viewing transformation. Otherwise, no other transformation matrices are affected.

To ensure correct handling of text size and tracking relationships, *mapping_mode* should be called prior to setting text size and tracking. This procedure also may change the number of passes required for hidden surface removal, so hidden surface removal should be disabled while setting the mapping mode.

DEFAULTS

Isotropic

SEE ALSO

hidden_surface(3G), vdc_extent(3G), vdc_justification(3G), set_p1_p2(3G).

NAME

marker_color – select color for subsequent polymarker primitives

SYNOPSIS

C Syntax:

```
void marker_color_index(fildes,index);
int fildes,index;

void marker_color(fildes,red,green,blue);
int fildes;
float red,green,blue;
```

FORTRAN77 Syntax:

```
subroutine marker_color_index(fildes,index)
integer*4 fildes,index

subroutine marker_color(fildes,red,green,blue)
integer*4 fildes
real red,green,blue
```

Pascal Syntax:

```
procedure marker_color_index(fildes,index:integer);

procedure marker_color(fildes:integer;red,green,blue:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

index Integer index into the device color table. The marker color is the red, green, and blue values specified at this table location. The size of the color table can be obtained by using the *inquire_sizes* procedure.

red, green, blue Color values (in the range of 0.0 to 1.0) to be used for subsequent polymarker primitives. Zero (0.0) indicates **no** color and one (1.0) indicates **full** color. Monochrome devices use the equation

$$I = 0.30(\text{red}) + 0.59(\text{green}) + 0.11(\text{blue})$$

to determine the intensity.

Discussion

To acquire the current color table definition, use the *inquire_color_table* command. To change the current color table definition, use the *define_color_table* procedure.

Polymarker primitives are altered by this procedure.

When the marker color is set using red, green, and blue parameters, determination of actual output depends on the state of the mode parameter set by *shade_mode*:

CMAP_NORMAL — Default mode if *shade_mode* has not been called. The color table is searched for an index pointing to the color in RGB space that most closely matches the one specified. Specification by index is faster than color map searches.

CMAP_MONOTONIC — Color values are converted to intensity using the equation:

$$I = 0.30(\text{red}) + 0.59(\text{green}) + 0.11(\text{blue}).$$

This intensity is mapped to an index where minimum and maximum intensity is defined by *shade_range*.

CMAP_FULL — Color values are mapped directly to a index with the assumption that the color map is set up to a predefined full color state.

If the color map is changed after this procedure is called, the marker color used may not be the original color desired.

If the index is out of range, a warning is generated and a mod function is performed.

DEFAULTS

Default marker color is the color defined in the current color map at index 1.

SEE ALSO

define_color_table(3G), inquire_color_table(3G), inquire_sizes(3G), polymarker(3G), shade_mode(3G).

NAME

marker_orientation – define orientation of symbols drawn with marker primitives

SYNOPSIS

C Syntax:

```
void marker_orientation(fildes,x,y);
int fildes;
float x,y;
```

FORTRAN77 Syntax:

```
subroutine marker_orientation(fildes,x,y)
integer*4 fildes
real x,y
```

Pascal Syntax:

```
procedure marker_orientation(fildes:integer;x,y:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

x,y Components of a vector relative to (0.0,0.0) that determines the orientation of subsequent marker symbols. Vector is specified in vdc coordinates.

DISCUSSION

The *marker_orientation* vector defines the slant of marker symbols and is not a rotation. That is, the orientation vector determines the movement of subsequent marker symbols away from the vertical.

DEFAULTS

x = 0.0; **y** = 1.0.

NAME

marker_size – select polymarker size

SYNOPSIS

C Syntax:

```
void marker_size(fildes,size,mode);
int fildes,mode;
float size;

void dcmarker_size(fildes,dcsize);
int fildes,dcsize;
```

FORTRAN77 Syntax:

```
subroutine marker_size(fildes,size,mode)
integer*4 fildes,mode
real size

subroutine dcmarker_size(fildes,dcsize)
integer*4 fildes,dcsize
```

Pascal Syntax:

```
procedure marker_size(fildes:integer;size:real;
        mode:integer);

procedure dcmarker_size(fildes,dcsize:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

size Marker height measured in virtual device coordinate units along the marker up vector, or a scaling factor (see **mode** below).

mode If **mode** = FALSE, the size is an absolute measure in virtual device coordinate units. If size is zero, the smallest discernable marker size is used. If **mode** = TRUE, a scaling factor is applied to a device-dependent marker size.

dcsize Polymarker height measured in device coordinate units.

DEFAULTS

marker size = 0.
mode = FALSE (the smallest discernable marker).
dcsize = 10 device coordinate units.

NAME

marker_type – select marker type for subsequent marker primitives

SYNOPSIS

C Syntax:

```
void marker_type(fildes,marker);
int fildes,marker;
```

FORTRAN77 Syntax:

```
subroutine marker_type(fildes,marker)
integer*4 fildes,marker
```

Pascal Syntax:

```
procedure marker_type(fildes,marker:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

marker An enumerated value of:

0 = dot	10 = one
1 = plus	11 = two
2 = asterisk	12 = three
3 = circle	13 = four
4 = cross	14 = five
5 = triangle	15 = six
6 = square	16 = seven
7 = diamond	17 = eight
8 = square with cross	18 = nine
9 = zero	

Discussion

All markers listed above are centered about the points specified in the *polymarker2d* or *polymarker3d* procedure.

An out-of-range marker, either less than zero or greater than 18, leaves the old marker specification intact.

DEFAULTS

2 (asterisk)

NAME

move2d, move3d, dcmove, intmove2d – update current pen position and move physical pen to that location

SYNOPSIS

C Syntax:

```
void move2d(fildes,x,y);
int fildes;
float x,y;

void move3d(fildes,x,y,z);
int fildes;
float x,y,z;

void dcmove(fildes,dcx,dcy);
int fildes,dcx,dcy;

void intmove2d(fildes,x,y);
int fildes,x,y;
```

FORTRAN77 Syntax:

```
subroutine move2d(fildes,x,y)
integer*4 fildes
real x,y

subroutine move3d(fildes,x,y,z)
integer*4 fildes
real x,y,z

subroutine dcmove(fildes,dcx,dcy)
integer*4 fildes,dcx,dcy

subroutine intmove2d(fildes,x,y)
integer*4 fildes,x,y
```

Pascal Syntax:

```
procedure move2d(fildes:integer;x,y:real);

procedure move3d(fildes:integer;x,y,z:real);

procedure dcmove(fildes,dcx,dcy:integer);

procedure intmove2d(fildes,x,y:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when a path to a graphics device is opened.

x,y,z Defines the position to which the current pen position is moved (updated) in world coordinate values.

dcx,dcy Defines the position to which the current pen position is moved (updated) in device coordinate values.

Discussion

Move2d, intmove2d, and *move3d* update the current pen position in world coordinates to the specified position. If the device identified by **fildes** has a physical pen, the physical pen is also relocated to the current pen position.

Dcmove performs an equivalent operation in device coordinates, but does not perform any transformation or clipping steps.

World coordinate current pen position is completely independent from device coordinate pen position. If you are switching between the two coordinate systems, be sure to do a *move* as the first procedure after changing coordinate systems before doing any other work with the new coordinates.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase(3G)* manual page.

For increased performance for *move2d* and *move3d*, macros can be used to generate in-line code for C. See "Move and Draw Macros" in *Starbase Graphics Techniques – HP-UX Concepts and Tutorials.*

SEE ALSO

draw(3G), inquire_current_position(3G), *Starbase Graphics Techniques.*

NAME

pattern_define – define a fill pattern

SYNOPSIS

C Syntax:

```
void pattern_define(fildes,dx,dy,pattern);
int fildes,dx,dy;
unsigned char pattern[];
```

FORTRAN77 Syntax:

```
subroutine pattern_define(fildes,dx,dy,pattern)
integer*4 fildes,dx,dy
character*(*) pattern()
```

Pascal Syntax:

```
type
        gbyte = 0..255;

procedure pattern_define(fildes,dx,dy:integer;pattern:packed
array[lo..hi:integer] of gbyte);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

dx,dy The number of pixels defined in the x, and y directions, respectively. The maximum value for each is 16, and both dx and dy must be powers of two (i.e. 1, 2, 4, 8, or 16).

pattern The fill pattern. Each element of the array (up to 8 bits) represents one pixel in the pattern. The first element of the array corresponds to the upper left corner of the pattern. The next pixel (and second element of the array) is immediately to its right (or directly below if the pattern is only one column wide). The last pixel (and last element of the array) is the lower right corner (or bottom pixel if the pattern is only one column wide).

Discussion

A **dx** by **dy** pixel fill pattern can be defined by the user. The value of each element of the pattern array corresponds to the current color in that position of the color map.

For example, the following segment:

```
for (i=0; i<256; i++)
  pattern[i]=0x06;
pattern_define(fildes,16,16,pattern);
```

would set all the pixels to blue (assuming the default color map). The pattern is not used unless *interior_style* has been called with **style** = INT_PATTERN.

For hardware that does not support the full 16x16 cell size, the cell size will be device-dependent. See the *Starbase Device Drivers Library* manual for the specifics for a particular device.

The pattern is used for filling only when *shade_mode* is set to CMAP_NORMAL or CMAP_MONOTONIC.

Defaults

All elements in the pattern are set to white.

SEE ALSO

interior_style(3G), rgb_to_index(3G), shade_mode(3G).

NAME

perimeter_color, bf_perimeter_color – select color index or color value for subsequent polygon perimeters, or for backfacing polygon perimeters

SYNOPSIS

C Syntax:

```
void perimeter_color_index(fildes,index);
int fildes,index;

void perimeter_color(fildes,red,green,blue);
int fildes;
float red,green,blue;

void bf_perimeter_color_index(fildes,index);
int fildes,index;

void bf_perimeter_color(fildes,red,green,blue);
int fildes;
float red,green,blue;
```

FORTRAN77 Syntax:

```
subroutine perimeter_color_index(fildes,index)
integer*4 fildes,index

subroutine perimeter_color(fildes,red,green,blue)
integer*4 fildes
real red,green,blue

subroutine bf_perimeter_color_index(fildes,index)
integer*4 fildes,index

subroutine bf_perimeter_color(fildes,red,green,blue)
integer*4 fildes
real red,green,blue
```

Pascal Syntax:

```
procedure perimeter_color_index(fildes,index:integer);

procedure perimeter_color(fildes:integer;red,green,blue:real);

procedure bf_perimeter_color_index(fildes,index:integer);

procedure bf_perimeter_color(fildes:integer;red,green,blue:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

index Integer index into the device color table. The perimeter color, or the backfacing perimeter color, is the red, green, and blue values specified at this table location. The size of the color table can be obtained using the *inquire_sizes* procedure.

red, green, blue

Color values (in the range of 0.0 to 1.0) to be used for subsequent polygon perimeters, or backfacing polygon perimeters. Zero (0.0) indicates **no** color and one (1.0) indicates **full** color. To determine the intensity, monochrome devices use the equation:

$$I = 0.30 * red + 0.59 * green + 0.11 * blue$$

Discussion

Backfacing polygon edges use the specified perimeter color after *bf_control* has been called with **attr** parameter set to TRUE(1).

To acquire the current color table definition, use the *inquire_color_table* command. To change the current color table definition, use the *define_color_table* procedure.

Polygon primitives are altered by this procedure.

When either the perimeter color or the backfacing perimeter color is set using red, green, and blue parameters, determination of actual output depends on the state of the mode parameter set with *shade_mode*:

CMAP_NORMAL — This is the default mode if *shade_mode* has not been called. The color table is searched for an index which points to the closest color in RGB space to the one specified. Specification by index is faster than color map searches.

CMAP_MONOTONIC — Color values are converted to intensity by using the equation:

$$I = 0.30 * red + 0.59 * green + 0.11 * blue$$

This intensity is mapped to an index using the minimum and maximum defined by *shade_range*.

CMAP_FULL — Color values are mapped directly to an index with the assumption that the color map is set up to a predefined full color state.

If the color map is changed after this procedure is called, the perimeter color, or the backfacing perimeter color, used may not be the original color desired. If the index is out of range, a warning is generated and a **mod** function is performed.

Light source calculations and color coordinates in the polygon vertex list do not affect the color of perimeter, or backfacing perimeter, edges.

DEFAULTS

Default perimeter color, and backfacing perimeter color, is the color defined in the current color map at index 1.

SEE ALSO

bf_control(3G), define_color_table(3G), fill_color(3G), inquire_color_table(3G), interior_style(3G), polygon(3G), shade_mode(3G), surface_coefficients(3G), surface_model(3G), *Starbase Graphics Techniques.*

NAME

perimeter_repeat_length, intperimeter_repeat_length, bf_perimeter_repeat_length – define line type pattern size for polygon perimeters and backfacing polygon perimeters

SYNOPSIS

C Syntax:

```
void perimeter_repeat_length(fildes,length);
int fildes;
float length;

void intperimeter_repeat_length(fildes,length);
int fildes,length;

void bf_perimeter_repeat_length(fildes,length);
int fildes;
float length;
```

FORTRAN77 Syntax:

```
subroutine perimeter_repeat_length(fildes,length)
integer*4 fildes
real length

subroutine intperimeter_repeat_length(fildes,length)
integer*4 fildes,length

subroutine bf_perimeter_repeat_length(fildes,length)
integer*4 fildes
real length
```

Pascal Syntax:

```
procedure perimeter_repeat_length(fildes:integer;length:real);

procedure intperimeter_repeat_length(fildes,length:integer);

procedure bf_perimeter_repeat_length(fildes:integer;length:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

length Repeat length of the perimeter type pattern, or backfacing perimeter type pattern, measured in virtual device units along the x-direction.

Discussion

Perimeter_repeat_length and *intperimeter_repeat_length* set the repeat length for *polygon*, *arc*, *circle*, *rectangle*, and *ellipse* output primitive perimeters.

bf_perimeter_repeat_length sets the repeat length for the perimeter of backfacing elements of *polygon* and *rectangle*.

There are some device-dependent limitations for setting repeat length. Some raster devices round to the nearest multiple of 16 pixels.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase(3G)* manual page.

DEFAULT

Repeat length is 1/32 of the default *vdc_extent* x-axis.

SEE ALSO

line_repeat_length(3G), perimeter_type(3G), vdc_extent(3G).

NAME

perimeter_type, bf_perimeter_type – select line type for subsequent polygon perimeters and backfacing polygon perimeters

SYNOPSIS

C Syntax:

```
void perimeter_type(fildes,style);
int fildes,style;

void bf_perimeter_type(fildes,style);
int fildes,style;
```

FORTRAN77 Syntax:

```
subroutine perimeter_type(fildes,style)
integer*4 fildes,style

subroutine bf_perimeter_type(fildes,style)
integer*4 fildes,style
```

Pascal Syntax:

```
procedure perimeter_type(fildes,style:integer);

procedure bf_perimeter_type(fildes,style:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

style One of the following:

SOLID
DASH
DOT
DASH_DOT
DASH_DOT_DOT
LONG_DASH
CENTER_DASH
CENTER_DASH_DASH

Discussion

The following perimeter types, and backfacing perimeter types, are pre-defined:

Index	Name	Approximate Pattern
0	SOLID	Solid
1	DASH	.25 .50 .25
2	DOT	4-8 dots per repeat length
3	DASH_DOT	.4 .1 dot .1 .35
4	DASH_DOT_DOT	.35 .1 dot .1 dot .1 .35
5	LONG_DASH	.375 .25 .375
6	CENTER_DASH	.35 .1 .1 .1 .35
7	CENTER_DASH_DASH	.25 .1 .1 .1 .1 .1 .25

Frontfacing polygon edges use the perimeter type specified by *perimeter_type*. Backfacing polygon edges use the perimeter type specified by *bf_perimeter_type* after *bf_control* has been called with **attr** parameter set to TRUE(1).

Negative **style** values are device-dependent. For example, −1 is a point plot on a terminal.

Continuity between separate but graphically connected polyline or line segments is not guaranteed.

The pattern is restarted each time this procedure is executed.

Some devices do not support this procedure.

DEFAULTS

Solid Lines.

SEE ALSO

line_type(3G), perimeter_repeat_length(3G), *Starbase Device Drivers Library Manual.*

NAME

pick_depth – define pick depth for *pick_from_segment*(3G) and *hit_mode*(3G)

SYNOPSIS

C Syntax:

```
void set_pick_depth(fildes,p_front,p_back)
int fildes;
float p_front,p_back;

void inq_pick_depth(fildes,p_front,p_back)
int fildes;
float *p_front,*p_back;
```

FORTRAN77 Syntax:

```
subroutine set_pick_depth(fildes,p_front,p_back)
integer*4 fildes
real p_front,p_back

subroutine inq_pick_depth(fildes,p_front,p_back)
integer*4 fildes
real p_front,p_back
```

Pascal Syntax:

```
procedure set_pick_depth(fildes:integer; p_front,p_back:real);

procedure inq_pick_depth(fildes:integer; var p_front,p_back:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

p_front Specifies the front pick aperture plane in perspective coordinates; i.e., comparison with the pick aperture is performed after the primitives are transformed by the current transformation matrix. Perspective divide occurs after the front and back clip.

p_back Specifies the back pick aperture plane in perspective coordinates; i.e., comparison with the pick aperture performed after the primitives are transformed by the current transformation matrix. Perspective divide occurs after the front and back clip.

Discussion

Set_pick_depth sets a front and back *z* plane which together with *pick_window* defines the pick aperture. *Inq_pick_depth* returns the current set values for the pick depth.

The pick depth is used for hit testing when the Starbase hit mode is enabled (see *hit_mode*(3G)). Hit testing is also done by *pick_from_segment*(3G).

Some picking modes (see *set_pick_mode*(3G) in the *Starbase Display List Programmers Manual*) may cause *pick_from_segment*(3G) to alter the pick depth.

ERRORS

1 Graphics device is not initialized for this operation.

DEFAULTS

p_front: −10 000.0
p_back: 10 000.0

SEE ALSO

hit_mode(3G), pick_from_segment(3G), pick_window(3G).

NAME

intset_pick_window, intinquire_pick_window, inq_pick_window, set_pick_window – define pick window for *pick_from_segment*(3G) and *hit_mode*(3G)

SYNOPSIS

C Syntax:

```
void intinquire_pick_window(fildes,px_min,py_min,px_max,py_max)
int fildes,*px_min,*py_min,*px_max,*py_max;

void intset_pick_window(fildes,px_min,py_min,px_max,py_max)
int fildes,px_min,py_min,px_max,py_max;

void inq_pick_window(fildes,px_min,py_min,px_max,py_max)
int fildes;
float *px_min,*py_min,*px_max,*py_max;

void set_pick_window(fildes,px_min,py_min,px_max,py_max)
int fildes;
float px_min,py_min,px_max,py_max;
```

FORTRAN77 Syntax:

```
subroutine intinquire_pick_window(fildes,px_min,py_min,px_max,py_max)
integer*4 fildes,px_min,py_min,px_max,py_max

subroutine intset_pick_window(fildes,px_min,py_min,px_max,py_max)
integer*4 fildes,px_min,py_min,px_max,py_max

subroutine inq_pick_window(fildes,px_min,py_min,px_max,py_max)
integer*4 fildes
real px_min,py_min,px_max,py_max

subroutine set_pick_window(fildes,px_min,py_min,px_max,py_max)
integer*4 fildes
real px_min,py_min,px_max,py_max
```

Pascal Syntax:

```
procedure intinquire_pick_window(fildes:integer; var px_min,py_min,px_max,
        py_max:integer);

procedure intset_pick_window(fildes,px_min,py_min,px_max,py_max:integer);

procedure inq_pick_window(fildes:integer;var px_min,py_min,px_max,py_max:real);

procedure set_pick_window(fildes:integer;px_min,py_min,px_max,py_max:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

px_min minimum x limit for pick window in VDC units.

py_min minimum y limit for pick window in VDC units.

px_max maximum x limit for pick window in VDC units.

py_max maximum y limit for pick window in VDC units.

Discussion

Set_pick_window and *intset_pick_window* set a 2-dimensional rectangular window which together with *pick_depth* defines a pick aperture.

Inq_pick_window and *intinquire_pick_window* returns the current set values for the pick window.

The pick window is used for hit testing when the Starbase hit mode is enabled (see *hit_mode*(3G)). Hit testing is also done by *pick_from_segment*(3G).

Note that some picking modes (see *set_pick_mode*(3G) in the *Starbase Display List Programmers Manual* may cause *pick_from_segment*(3G) to alter the pick window.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

ERRORS

1 Graphics device is not initialized for this operation.

DEFAULTS

Float: **(px_min,py_min,px_max,py_max)=(0.0,0.0,1.0,1.0)**
Integer: **(px_min,py_min,px_max,py_max)=(0,0,32767,32767)**

SEE ALSO

hit_mode(3G), pick_depth(3G), pick_from_segment(3G), set_pick_mode(3G).

NAME

intpolycircle, dcpolycircle – define circular regions to be filled and/or edged.

SYNOPSIS

C Syntax:

void dcpolycircle (fildes,clist,numcircles,radius,flags);
int fildes,clist[],numcircles,radius,flags;

void intpolycircle (fildes,clist,numcircles,radius,flags);
int fildes,clist[],numcircles,radius,flags;

FORTRAN77 Syntax:

subroutine dcpolycircle (fildes,clist,numcircles,radius,flags);
integer*4 fildes,clist(numcircles*(2+flags)),numcircles,radius,flags;

subroutine intpolycircle (fildes,clist,numcircles,radius,flags);
integer*4 fildes,clist(numcircles*(2+flags)),numcircles,radius,flags;

Pascal Syntax:

procedure dcpolycircle (fildes:integer,var clist:array[lo..hi:integer] of integer;
numcircles,radius,flags:integer);

procedure intpolycircle (fildes:integer,var clist:array[lo..hi:integer] of integer;
numcircles,radius,flags:integer);

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

clist Array of integer world coordinate data used by *intpolycircle* or integer device coordinate data used by *dcpolycircle*. Data is arranged in x, y pairs representing the circles' centers. This data may contain move/draw indicators after each x, y pair.

numcircles Number of circles to be drawn.

radius Radius of the circles, in integer world coordinates or device coordinates.

flags If set to FALSE (0), move/draw data is not included in the **clist**.

If set to TRUE (1), move/draw indicators are included in the **clist** following each pair of *x, y* coordinates. Move/draw indicators occupy the same amount of space as a single coordinate and are interspersed with the coordinate data. All bits of the indicator must be zero to indicate a move (integer 0). If move/draw indicators are present, each (x,y) pair of coordinates is followed by a move/draw indicator. For example:

2-dimensions
x1
y1
move/draw1
x2
y2
move/draw2
.
.
.
.

Discussion

Circles with the specified radius are drawn, centered at (x,y) as specified in the **clist** array, regardless of whether that coordinate contains a move or a draw flag. If **flags** is TRUE (1), the move/draw flag following the position coordinate is ignored.

Each entry in the clist may contain any number of coordinates. Note that *vertex_format* does not apply to intpolycircle.

The circles are filled, left hollow, and outlined according to the current interior style. As with all output primitives, the arcs are affected by the current drawing mode and write enable.

No clipping or transformations are performed on device coordinate procedures.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode floating point operations are not available for that file descriptor. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

NOTE

Doing any Starbase calls other than *intpartial_circle* or *intpartial_polygon* in the middle of a list of procedure calls that add vertices to the polygon buffer and before a call to *intpolygon*, *intrectangle*, *intarc*, *intcircle*, *intpolyrectangle*, or *intpolycircle* produces unpredictable, device-dependent results.

When using a distorted mapping (such as when the number of VDC units in the x direction does not equal the number of VDC units in the y direction, or the current transformation matrix scales the x and y directions differently), integer circles may not remain circular. This effect is unpredictable and device-dependent, and may not remain consistent with future releases.

SEE ALSO

circle(3G), drawing_mode(3G), fill_color(3G), vertex_format(3G), write_enable(3G), *Starbase Graphics Techniques.*

NAME

dcpartial_polygon, dcpolygon, intpartial_polygon2d, intpolygon2d, partial_polygon2d, partial_polygon3d, polygon2d, polygon3d – defines a polygonal region to be filled and/or edged, or defines a group of polygon vertices that begins as subpolygon or non-edged boundary.

SYNOPSIS

C Syntax:

```
void dcpartial_polygon(fildes,clist,numverts,flags,closure);
int fildes,numverts,flags,closure,clist[];

void dcpolygon(fildes,clist,numverts,flags);
int fildes,numverts,flags;
int clist[];

void intpartial_polygon2d(fildes,clist,numverts,flags,closure);
int fildes,numverts,flags,closure,clist[];

void intpolygon2d(fildes,clist,numverts,flags);
int fildes,numverts,flags,clist[];

void partial_polygon2d(fildes,clist,numverts,flags,closure);
int fildes,numverts,flags,closure;
float clist[];

void partial_polygon3d(fildes,clist,numverts,flags,closure);
int fildes, numverts,flags,closure;
float clist[];

void polygon2d(fildes,clist,numverts,flags);
int fildes,numverts,flags;
float clist[];

void polygon3d(fildes,clist,numverts,flags);
int fildes,numverts,flags;
float clist[];
```

FORTRAN77 Syntax:

```
subroutine dcpartial_polygon(fildes,clist,numverts,flags,closure)
integer*4 fildes,numverts,flags,closure
integer*4 clist(numverts*(2+flags))

subroutine dcpolygon(fildes,clist,numverts,flags)
integer*4 fildes,numverts,flags
integer*4 clist(numverts*(2+flags))

subroutine intpartial_polygon2d(fildes,clist,numverts,flags,closure)
integer*4 fildes,numverts,flags,closure,clist(numverts*(2+flags))

subroutine intpolygon2d(fildes,clist,numverts,flags)
integer*4 fildes,numverts,flags,clist(numverts*(2+flags))

subroutine partial_polygon2d(fildes,clist,numverts,flags,closure)
integer*4 fildes,numverts,flags,closure
real clist(numverts*(2+flags))

subroutine partial_polygon3d(fildes,clist,numverts,flags,closure)
integer*4 fildes,numverts,flags,closure
real clist(numverts*(3+flags))

subroutine polygon2d(fildes,clist,numverts,flags)
integer*4 fildes,numverts,flags
```

real clist(numverts*(2+flags))

subroutine polygon3d(fildes,clist,numverts,flags)
integer*4 fildes,numverts,flags
real clist(numverts*(3+flags))

Pascal Syntax:

procedure dcpartial_polygon(fildes:integer;var clist:array[lo..hi:integer] of integer; numverts,flags,closure:integer);

procedure dcpolygon(fildes:integer;var clist:array[lo..hi:integer] of integer; numverts,flags:integer);

procedure intpartial_polygon2d(fildes:integer;var clist:array[lo..hi:integer] of integer; numverts,flags,closure:integer);

procedure intpolygon2d(fildes:integer;var clist:array[lo..hi:integer] of integer; numverts,flags:integer);

procedure partial_polygon2d(fildes:integer;var clist:array[lo..hi:integer] of real; numverts,flags,closure:integer);

procedure partial_polygon3d(fildes:integer;var clist:array[lo..hi:integer] of real; numverts,flags,closure:integer);

procedure polygon2d(fildes:integer;var clist:array[lo..hi:integer] of real; numverts,flags:integer);

procedure polygon3d(fildes:integer;var clist:array[lo..hi:integer] of real; numverts,flags:integer);

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

flags *polygon:* If set to FALSE (0), all edges are drawn if *interior_style* is set to be edged.

partial_polygon: If FALSE (0), the first vertex represents a non-drawn boundary. The remaining vertices represent drawn boundaries if *interior_style* is set to be edged.

polygon and *partial_polygon:* If set to TRUE (1), each polygon edge can be either a drawn boundary or a non-drawn boundary, depending on the value of the corresponding move/draw indicator and whether or not *interior_style* is set to be edged. If the indicator is non-zero (draw) and *interior_style* is set to **edged**, the boundary is drawn. Otherwise, it is not drawn. Non-drawn boundaries simply define the edges of the filled area.

clist Array of real or integer data with or without move/draw indicators embedded after each endpoint.

closure *partial_polygon:* If TRUE (1), the polygon vertex list is non-empty, and the last subpolygon is not yet closed, a boundary is added to close that subpolygon within the boundary. If the first polygon vertex in the partial polygon had a draw indicator, the closure boundary is drawn. Otherwise, it is not drawn. *Partial_polygon* then adds the vertices in **clist** to the polygon vertex list.

numverts Number of polygon vertices in the clist array. If the value is less than zero, zero is substituted.

Discussion

A boundary of a polygonal region is defined by connecting each vertex to its successor in the vertex list. The polygon is filled and/or outlined according to current interior style, fill color and perimeter attributes. As with all output primitives, it is affected by the current drawing mode and write enable.

Move/draw indicators occupy the same space as one coordinate (a single 32-bit value), and are interspersed with the coordinate data. For *polygon2d*, *partial_polygon2d*, *intpolygon2d*, *intpartial_polygon2d*, *dcpartial_polygon* and *dcpolygon* each (x,y) pair of coordinates is followed by a move/draw indicator if **flags** is TRUE(1). For *polygon3d* and *partial_polygon3d* each (x,y,z) triplet is followed by a move/draw indicator if **flags** is TRUE(1).

Partial_polygon puts polygon vertices into an internal polygon data structure. Only after a call is made to *polygon*, *circle*, *polycircle*, *arc*, *ellipse*, *rectangle*, or *polyrectangle* will the polygon be rendered. *Partial_polygon* is most commonly used to create "holes" in polygons.

Each entry in the **clist** can contain any number of coordinates. The actual number depends on the **coord** parameter specified in *vertex_format* if using device coordinates or floating point coordinates. *vertex_format* does not apply to *intpolygon2d* or *intpartial_polygon2d*. The **coord** parameter can be used to skip over any extra coordinates following the usual 2, 3, or 4 that can be specified with this procedure. Vertex move/draw flags follow the skipped extra coordinates.

The **use** parameter specified in *vertex_format* determines whether any of the extra parameters are used to determine the color of the primitive. The following list shows in what order the data is expected. Any extra coordinates are skipped.

use=0	For 2d,dc: x,y,...,(flag) For 3d: x,y,z,...,(flag)
use=1	For 2d,dc: x,y,i,...,(flag) For 3d: x,y,z,i,...,(flag)
use=3	For 2d,dc: x,y,a,b,c,...,(flag) For 3d: x,y,z,a,b,c,...,(flag)

All extra parameters are ignored by some devices.

Upon entry, the polygon vertex list is added to, not cleared. The polygon list is then processed by filling all vertices in the list, and cleared upon exit. The flag at the beginning (i.e., the first entry) of the **clist** for *partial_polygon* is used to create sub-polygons.

The current position is updated to the first vertex in the first partial_polygon/polygon upon completion of the polygon.

If per-polygon normal is set in *vertex_format*, the first vertex in *polygon3d* is used as a normal to that polygon, and the number of vertices in the **clist** should be **numverts** + 1. The normal must contain the same number of coordinates as every other vertex (extra coordinates are ignored).

If **normals** per polygon is specified in *vertex_format*, the normal is contained in the *polygon* vertex list; not in the *partial_polygon* list. If a normal needs to be calculated, the first 3 points of the first *partial_polygon* are used. The vertex list for the first partial polygon must contain at least 3 points or the normal cannot be calculated correctly.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

NOTE

Any Starbase call other than *intpartial_polygon2d*, *intpartial_arc*, *intpartial_circle*, *partial_polygon2d*, *partial_polygon3d*, *partial_arc*, *partial_ellipse*, or *dcpartial_polygon*, in the middle of a list of these procedure calls and before a call to *arc*, *circle*, *polycircle*, *ellipse*, *polygon*, *rectangle*, or *polycircle* produces unpredictable, device-dependent results.

SEE ALSO

drawing_mode(3G), fill_color(3G), interior_style(3G), perimeter_color(3G), perimeter_repeat_length(3G), perimeter_type(3G), write_enable(3G), vertex_format(3G).

NAME

dcpolyline, intpolyline2d, polyline2d, polyline3d – move/draw between specified points

SYNOPSIS

C Syntax:

```
void dcpolyline(fildes,clist,numpts,flags);
int fildes,numpts,flags,clist[];

void intpolyline2d(fildes,clist,numpts,flags);
int fildes,numpts,flags,clist[];

void polyline2d(fildes,clist,numpts,flags);
int fildes,numpts,flags;
float clist[];

void polyline3d(fildes,clist,numpts,flags);
int fildes,numpts,flags;
float clist[];
```

FORTRAN77 Syntax:

```
subroutine dcpolyline(fildes,clist,numpts,flags)
integer*4 fildes,numpts,flags,clist(numpts*(2+flags))

subroutine intpolyline2d(fildes,clist,numpts,flags)
integer*4 fildes,numpts,flags,clist(numpts*(2+flags))

subroutine polyline2d(fildes,clist,numpts,flags)
integer*4 fildes,numpts,flags
real clist(numpts*(2+flags))

subroutine polyline3d(fildes,clist,numpts,flags)
integer*4 fildes,numpts,flags
real clist(numpts*(3+flags))
```

Pascal Syntax:

```
procedure dcpolyline(fildes:integer;var clist:array[lo..hi:integer]
of integer;numpts,flags:integer);

procedure intpolyline2d(fildes:integer;var clist:array[lo..hi:integer]
of integer;numpts,flags:integer);

procedure polyline2d(fildes:integer;var clist:array[lo..hi:integer]
of real;numpts,flags:integer);

procedure polyline3d(fildes:integer;var clist:array[lo..hi:integer]
of real;numpts,flags:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

clist Array of virtual device coordinate data, modelling coordinate data, or world coordinate data used by *polyline2d*, *intpolyline2d*, and *polyline3d* or device coordinate data used by *dcpolyline*. This data may contain extra coordinates and move/draw indicators embedded after each endpoint.

numpts Defines the number of endpoints in the **clist** array. If **numpts** is less than zero, zero is substituted. A **numpts** greater than 32767 is not allowed.

flags If set to FALSE (0), the first endpoint represents a move and subsequent endpoints represent draws.

If set to TRUE (1), the move/draw indicators define whether a move or draw is performed for the corresponding endpoints. For each endpoint, if the corresponding move/draw indicator is non-zero (draw), a line is drawn from the current position to the endpoint. Otherwise, the current position is simply updated to the position of the endpoint.

Discussion

Polyline draws a series of line segments in the current *line_color* and *line_type*. A color can be specified for each endpoint, and the specified color can override *line_color* (see *vertex_format* for details). As with all output primitives, *polyline* is affected by the current *drawing_mode* and *write_enable*.

The line **type** pattern is continuous throughout a polyline.

Each entry in the **clist** can contain any number of coordinates. The actual number depends on the **coord** parameter specified in *vertex_format* . The **coord** parameter can be used to skip over any extra coordinates following the usual 2, 3, or 4 that can be specified with this procedure. Vertex move/draw flags follow the skipped extra coordinates.

The **use** parameter specified in *vertex_format* determines whether any of the extra parameters are used to determine the color of the primitive. (*vertex_format* does not apply to *intpolyline2d*.) The following shows what order the data is expected. Any extra coordinates are skipped.

use=0	For 2d, dc,: x,y,...,(flag) For 3d: x,y,z,...,(flag)
use=1	For 2d, dc,: x,y,i,...,(flag) For 3d: x,y,z,i,...,(flag)
use=3	For 2d, dc,: x,y,a,b,c,...,(flag) For 3d: x,y,z,a,b,c,...,(flag)

The world-coordinate current pen position is not related to the device current pen position. When mixing the use of these coordinate systems, always make a *move* to a known pen position before executing a *draw* procedure.

All extra parameters may be ignored by some devices.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase(3G)* manual page.

For increased performance of small polylines (**numpts** < 10) macros can be used to generate in-line code for C. See "Polyline Macros" in *Starbase Graphics Techniques – HP-UX Concepts and Tutorials.*

SEE ALSO

draw(3G), inquire_current_position(3G), line_color(3G), line_endpoint(3G), line_repeat_length(3G), line_type(3G), line_width(3G), move(3G), vertex_format(3G), *Starbase Graphics Techniques.*

NAME

polymarker2d, polymarker3d, dcpolymarker – draw current marker symbol centered at each endpoint specified in clist array

SYNOPSIS

C Syntax:

```
void polymarker2d(fildes,clist,numpts,flags);
int fildes,flags,numpts;
float clist[];

void polymarker3d(fildes,clist,numpts,flags);
int fildes,flags,numpts;
float clist[];

void dcpolymarker(fildes, clist, numpts, flags);
int fildes,flags,numpts,clist[];
```

FORTRAN77 Syntax:

```
subroutine polymarker2d(fildes,clist,numpts,flags)
integer*4 fildes,flags,numpts
real clist(numpts*(2+flags))

subroutine polymarker3d(fildes,clist,numpts,flags)
integer*4 fildes,flags,numpts
real clist(numpts*(3+flags))

subroutine dcpolymarker(fildes,clist,numpts,flags)
integer*4 fildes,flags,numpts
integer*4 clist(numpts*(2+flags))
```

Pascal Syntax:

```
procedure polymarker2d(fildes:integer;
        var clist:array[lo..hi:integer] of real;
        numpts,flags:integer);

procedure polymarker3d(fildes:integer;
        var clist:array[lo..hi:integer] of real;
        numpts,flags:integer);

procedure dcpolymarker(fildes:integer;
        var clist:array[lo..hi:integer] of integer;
        numpts,flags:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

clist Array of world (for *polymarker2d* and *polymarker3d*) or device (for *dcpolymarker*) coordinate data with or without extra coordinates and move/draw indicators embedded after each endpoint.

numpts Number of points in the **clist** array. If less than zero, zero is used.

flags If set to TRUE (1), move/draw data is included in the **clist** after *x,y,z* and any extra coordinates.

If set to FALSE (0), move/draw data is not included in the **clist**. Move/draw indicators occupy the same amount of space as a single coordinate and are interspersed with the coordinate data. All bits of the indicator must be zero to indicate a move (integer 0 or real 0.0). If the data is 2d, each (x,y) pair of coordinates is followed by a move draw indicator. If the data is 3d, each (x,y,z) triplet is followed by a move/draw indicator. For example:

2-dimensions	3-dimensions
x1	x1
y1	y1
m/d1	z1
x2	m/d1
y2	x2
m/d2	y2
.	z2
.	m/d2
.	.
.	.

Discussion

A marker is centered at each coordinate specified in the **clist** array regardless of whether that coordinate is a move or a draw. If flags is TRUE (1), the move/draw flag following the position coordinates is ignored.

Each entry in the clist may contain any number of coordinates. The actual number depends on the **coord** parameter specified in *vertex_format*. The **coord** parameter can be used to skip over any extra coordinates following the usual 2, 3, or 4 that can be specified with this procedure. Vertex move/draw flags follow the skipped extra coordinates.

The current marker symbol is drawn at each endpoint, independent of the move/draw indicators. Lines are not drawn between marker symbols. To draw lines between marker symbols, use the corresponding polyline procedure.

The marker is drawn with the current *marker_color*. As with all output primitives, *polymarker* is affected by the current *drawing_mode* and *write_enable*.

SEE ALSO

marker_color(3G), marker_orientation(3G), marker_size(3G), marker_type(3G), vertex_format(3G).

NAME

dcpolyrectangle, intpolyrectangle – define rectangular regions to be filled and/or edged

SYNOPSIS

C Syntax:

```
void intpolyrectangle(fildes,clist,numrects,flags);
int fildes,clist[],numrects,flags;

void dcpolyrectangle(fildes,clist,numrects,flags);
int fildes,clist[],numrects,flags;
```

FORTRAN77 Syntax:

```
subroutine intpolyrectangle(fildes,clist,numrects,flags)
integer*4 fildes,clist(numrects*(4+2*flags)),numrects,flags

subroutine dcpolyrectangle(fildes,clist,numrects,flags)
integer*4 fildes,clist(numrects*(4+2*flags)),numrect,flags;
```

Pascal Syntax:

```
procedure intrectangle(fildes:integer,var clist:array[lo..hi:integer] of integer;numrects,flags:inte
procedure dcrectangle(fildes:integer,var clist:array[lo..hi:integer] of integer;numrects,flags:integ
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

clist Array of integer world coordinate data for *intpolyrectangle* and integer device coordinate data for *dcpolyrectangle*. The data represents the corners of the rectangles, and should be arranged in the order x1, y1, x2, y2. This data may contain move/draw indicators after each x, y pair.

numrects The number of rectangles to be drawn.

flags If set to FALSE (0), move/draw data is not included in the **clist**.

If set to TRUE (1), move/draw indicators are included in the **clist** following each pair of *x, y* coordinates. Move/draw indicators occupy the same amount of space as a single coordinate and are interspersed with the coordinate data. All bits of the indicator must be zero to indicate a move (integer 0). If move/draw indicators are present, each (x,y) pair of coordinates is followed by a move/draw indicator. For example:

2-dimensions

x1
y1
move/draw1
x2
y2
move/draw2
.
.
.
.

Discussion

A boundary of a rectangular region is defined by diagonally opposite corners of the rectangle (x1, y1, x2, y2) as specified in the **clist** array, regardless of whether that coordinate contains a move or a draw flag. If **flags** is TRUE (1), the move/draw flag following the position

coordinate is ignored.

Each entry in the clist may contain any number of coordinates. Note that *vertex_format* does not apply to intpolyrectangle.

The rectangle is filled and/or edged according to the current interior style, using the current fill color and perimeter attributes. As with all output primitives, the rectangles are affected by the current drawing mode and write enable.

No clipping or transformations are performed on device coordinate procedures.

Device coordinate current pen position and world device coordinate current pen position are not related. If you are switching between the two coordinate systems, always begin the use of either system with a *move* to a known location before performing any other operations.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that file descriptor. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

NOTE

Doing any Starbase calls other than *intpartial_circle* or *intpartial_polygon* in the middle of a list of procedure calls that add vertices to the polygon buffer and before a call to *intpolygon*, *intrectangle*, *intarc*, *intcircle*, *intpolyrectangle*, or *intpolycircle* produces unpredictable, device-dependent results.

SEE ALSO

drawing_mode(3G), fill_color(3G), interior_style(3G), perimeter_color(3G), perimeter_repeat_length(3G), perimeter_type(3G), polygon(3G), rectangle(3G), vertex_format(3G), write_enable(3G).

NAME

intpop_matrix2d, pop_matrix, pop_matrix2d, pop_matrix3d – remove matrix from top of matrix stack

SYNOPSIS

C Syntax:

```
void intpop_matrix2d(fildes,xform2,radix,raw);
int fildes,*radix,raw;
int xform2[3][2];

void pop_matrix(fildes);
int fildes;

void pop_matrix2d(fildes,xform2);
int fildes;
float xform2[3][2];

void pop_matrix3d(fildes,xform3);
int fildes;
float xform3[4][4];
```

FORTRAN77 Syntax:

See Language Dependencies Below

```
subroutine intpop_matrix2d(fildes,xform2,radix,raw)
integer*4 fildes,radix,raw
real xform2(2,3)

subroutine pop_matrix(fildes)
integer*4 fildes

subroutine pop_matrix2d(fildes,xform2)
integer*4 fildes
real xform2(2,3)

subroutine pop_matrix3d(fildes,xform3)
integer*4 fildes
real xform3(4,4)
```

Pascal Syntax:

```
type
    int2d_xform = array [1..3][1..2] of integer;
    two_d_xform = array [1..3][1..2] of real;
    three_d_xform = array [1..4][1..4] of real;

procedure intpop_matrix2d(fildes:integer;var xform2:int2d_xform;
    var radix:integer;raw:integer);

procedure pop_matrix(fildes:integer);

procedure pop_matrix2d(fildes:integer;var xform:two_d_xform);

procedure pop_matrix3d(fildes:integer;var xform:three_d_xform);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

raw If set to TRUE(1), integer matrices will be in internal (raw) format, (i.e., translation values are not scaled).
If set to FALSE(0), all integer matrix values are scaled.

Output Parameters

xform2 3x2 (2-Dimensional) matrix.

xform3 4x4 (3-Dimensional) matrix.

radix is the radix factor for the 3x2 integer matrix.

Discussion

The top matrix on the matrix stack is called the *current transformation matrix.*

The *pop_matrix* procedure removes the top matrix from the matrix stack and throws it away.

The *pop_matrix2d, intpop_matrix2d,* and *pop_matrix3d* procedures are used to remove the top matrix from the matrix stack and return it in either *xform2* or *xform3*. If in MODEL_XFORM mode (see *gopen*), the matrix is returned as is. Otherwise, the popped matrix is concatenated with the inverse of the current viewing transformation. This inverse calculation is numerically intensive and will be done the first time a Starbase function needs the inverse of the viewing matrix after changing the viewing matrix. After calling one of these functions, the new current transformation matrix is the next matrix on the matrix stack.

The vdc-to-device unit transformation matrix and the viewing transformation matrix cannot be popped or replaced using matrix functions. The vdc matrix can be changed using *intvdc_extent, vdc_extent, vdc_justification, set_p1_p2,* or *mapping_mode*. The viewing matrix can be changed using *intview_matrix2d, intview_port, intview_window, view_camera, view_matrix, view_volume,* or *view_window*. If the matrix stack is empty, *pop_matrix* generates a warning.

Starbase maintains all internal floating point matrices in three-dimensional (4x4) form, so *pop_matrix2d* must shrink the current transformation matrix down before returning it.

Integer transformation matrices are scaled to allow a fractional portion for rotating objects. The radix factor indicates the number of bits to the right of the decimal point. Legal limits are 0 to 30. Once a coordinate has been transformed, it is divided by 2****radix** to return to an integer value.

When using raw mode with an integer matrix, positions (3,1) and (3,2) have an implied radix factor of 0 (no scaling). This allows large translation ranges along with accurate rotations.

Integer operations are only available when using the INT_XFORM *gopen* **mode**. When in INT_XFORM **mode**, floating point operations are not available for that **fildes**. Floating point operations are the default, or can be specified with FLOAT_XFORM **mode**. For a list of integer operations, floating point operations and common operations see the *starbase(3G)* manual page.

If matrices need to be read back without popping them off the matrix stack, use *read_matrices.*

LANGUAGE DEPENDENCIES

FORTRAN77:

A transposition of array rows and columns is required due to the manner in which FORTRAN77 stores arrays.

DEFAULTS

After *gopen*, the current transformation matrix is the vdc-to-device unit transformation matrix.

SEE ALSO

concat_matrix(3G), gopen(3G), mapping_mode(3G), push_matrix(3G), read_matrices(3G), set_p1_p2(3G), vdc_extent(3G), vdc_justification(3G), view_matrix(3G).

NAME

push_matrix2d, push_matrix3d, intpush_matrix2d – push matrix onto top of matrix stack

SYNOPSIS

C Syntax:

```
void push_matrix2d(fildes,xform2);
int fildes;
float xform2[3][2];

void push_matrix3d(fildes,xform3);
int fildes;
float xform3[4][4];

void intpush_matrix2d(fildes,xform2,radix,raw);
int fildes,xform2[3][2],radix,raw;
```

FORTRAN77 Syntax:

See Language Dependencies Below

```
subroutine push_matrix2d(fildes,xform2)
integer*4 fildes
real xform2(2,3)

subroutine push_matrix3d(fildes,xform3)
integer*4 fildes
real xform3(4,4)

subroutine intpush_matrix2d(fildes,xform2,radix,raw)
integer*4 fildes,xform2(2,3),radix,raw
```

Pascal Syntax:

```
type
int2d_xform = array [1..3][1..2] of integer;
two_d_xform = array [1..3][1..2] of real;
three_d_xform = array [1..4][1..4] of real;

procedure push_matrix2d(fildes:integer;
var xform2:two_d_xform);

procedure push_matrix3d(fildes:integer;
var xform2:three_d_xform);

procedure intpush_matrix2d(fildes:integer;
var xform2:int2d_xform;
radix,raw:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

xform2 3x2 (2-Dimensional) matrix.

xform3 4x4 (3-Dimensional) matrix.

radix is the radix factor for the 3x2 integer matrix.

raw If set to TRUE(1), integer matrices will be in internal (raw) format (i.e. translation values are not scaled).
If set to FALSE(0), all integer matrix values are scaled.

Discussion

The current transformation matrix (on top of matrix stack) is pushed to the second matrix

position on the matrix stack. If the graphics device has been opened in MODEL_XFORM mode, the matrix parameter, *xform2* or *xform3*, is placed on the top of the matrix stack. If the device is not in MODEL_XFORM mode, the matrix parameter is concatenated with the current viewing transformation matrix and the result is placed on the top of the matrix stack. The new transformation matrix is used to transform subsequent output primitives.

The coordinate systems used by Starbase can be conceptually defined as follows:

1. User points are assumed to be defined in modelling coordinates. These points are transformed into world coordinates using a matrix called the *modelling transformation matrix* (if any matrices have been pushed on the stack). World coordinates are used to perform any necessary rendering calculations. Use *push_matrix* to place modelling transformations on the matrix stack after defining the viewing transformation.
2. World coordinates are transformed to device coordinates by the *viewing transformation matrix* which is usually the concatenation of user-defined viewing transformations and the vdc-to-device transformation matrix. When a device is opened, the *viewing transformation matrix* is simply the vdc-to-device coordinate transformation matrix. The user can define further viewing transformations (such as perspective) with *intview_window, intview_port, intview_matrix2d view_camera, view_matrix, view_port, view_volume*, or *view_window*.

If a graphics device has been opened in MODEL_XFORM mode, special rendering calculations such as shading may be needed after the modelling-to-world-coordinate transformation. Therefore the modelling transformation cannot be combined with the viewing transformation. Thus, matrices pushed on the stack are left as is; all transformations from modelling coordinates to device coordinates occur in two steps: modelling-to-world-coordinates, followed by world-to-device-coordinates.

If a graphics device is not in MODEL_XFORM mode, the modelling and viewing transformations can be combined: The current viewing transformation matrix is post-concatenated to modelling matrices placed on the matrix stack, while subsequent output primitives are transformed using only the top matrix on the matrix stack.

Starbase maintains all internal floating point matrices in 3D (4x4) form, so *push_matrix2d* must expand the 2d matrix **xform2.**

Perspective transformations (as well as any other non-linear or viewing transformation) should not appear in modelling matrices because they distort surface normals and invalidate lighting calculations. Rather, they should appear in the viewing transformation which is set with *view_matrix*. When performing perspective transformations, a perspective model where the eye is at origin of perspective space is recommended. Any other model can easily be modified by a single translation step.

Integer transformation matrices are scaled to allow a fractional portion for rotating objects. The radix factor indicates the number of bits to the right of the decimal point. Legal limits are 0 to 30. Once a coordinate has been transformed, it is divided by 2****radix** to return to an integer value.

When using raw mode with an integer matrix, positions (3,1) and (3,2) have an implied radix factor of 0(no scaling). This allows large translation ranges along with accurate rotations.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

LANGUAGE DEPENDENCIES

FORTRAN77: FORTRAN requires a transposition of array rows and columns due to the manner in which FORTRAN77 stores arrays.

DEFAULTS

After *gopen*, the current transformation matrix is the vdc-to-device units transformation matrix.

SEE ALSO

concat_matrix(3G), concat_transformation(3G), pop_matrix(3G), vdc_extent(3G), view_matrix(3G).

NAME

push_vdc_matrix – push vdc-to-device units transformation matrix onto top of matrix stack.

SYNOPSIS

C Syntax:

```
void push_vdc_matrix(fildes);
int fildes;
```

FORTRAN77 Syntax:

```
subroutine push_vdc_matrix(fildes)
integer*4 fildes
```

Pascal Syntax:

```
procedure push_vdc_matrix(fildes:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

Discussion

A copy of the current vdc-to-device units transformation matrix is pushed onto the top of the matrix stack and becomes the current transformation matrix. If *vdc_extent*, *set_p1_p2*, *mapping_mode*, or *vdc_justification* changes, the internal Starbase vdc-to-device units transformation matrix changes, but not any vdc matrices that were pushed using push_vdc_matrix.

This routine should be used carefully (especially when a device has been opened in MODEL_XFORM mode), because only modelling matrices are normally pushed on the matrix stack. If the vdc-to-device matrix is pushed on the stack in MODEL_XFORM mode, lighting calculations for subsequent output primitives may be invalid. For lighting calculations in vdc coordinates, use *flush_matrices*.

SEE ALSO

push_matrix(3G), vdc_extent(3G), set_p1_p2(3G), mapping_mode(3G), vdc_justification(3G).

NAME

quadrilateral_mesh – defines a series of quadrilateral regions to be filled and/or edged.

SYNOPSIS

C Syntax:

```
void quadrilateral_mesh(fildes,clist,numverts_m,numverts_n,gnormals);
int fildes,numverts_m,numverts_n;
float clist[],gnormals[];
```

FORTRAN77 Syntax:

```
subroutine quadrilateral_mesh(fildes,clist,numverts_m,numverts_n,gnormals)
integer*4 fildes,numverts_m,numverts_n
real clist(numverts_m*numverts_n*3),gnormals((numverts_m-1)*(numverts_n-1)*3)
```

Pascal Syntax:

```
procedure quadrilateral_mesh(fildes:integer;var clist:array[lo..hi:integer] of real;
        numverts_m,numverts_n:integer;var gnormals:array[lo..hi:integer] of real);
```

DESCRIPTION

Input Parameters

fildes	Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.
clist	array of three dimensional real endpoint data stored in row major format
numverts_m	number of rows in the clist array.
numverts_n	number of columns in the clist array.
gnormals	array of real geometric normals.

Discussion

The list of points is a two dimensional array where each set of four neighboring points defines a quadrilateral. The first quadrilateral consists of the first two points from the first row and the first two points from the second row. The second quadrilateral consists of the second and third points from the first row and the second and third points from the second row and so on.

If *vertex_format* is specified to include a normal per polygon, then this normal is extracted from the *gnormals* array. The first three entries of this array are x,y,z coordinates of the normal for the first quadrilateral. The second three entries are the normal for the second quadrilateral if normals per vertex are present, and so on. If there are no normals per polygon then the gnormals value may be null.

The sense of the CLOCKWISE/COUNTER_CLOCKWISE direction of polygon vertices is determined by the 1st, 2nd and 3rd vertex in each quadrilateral.

For example: The 1st vertex in the 1st row is the 1st vertex of the first quadrilateral. The 2nd vertex in the 1st row is the 2nd vertex of the first quadrilateral. The 2nd vertex in the 2nd row is the 3rd vertex of the first quadrilateral.

Each quadrilateral is filled and/or outlined according to the current interior style. *Quadrilateral_mesh* uses the current fill color and perimeter attributes. As with all output primitives it is affected by the current drawing mode and write enable.

SEE ALSO

drawing_mode(3G), fill_color(3G), interior_style(3G), perimeter_color(3G), perimeter_repeat_length(3G), perimeter_type(3G), polygon(3G), vertex_format(3G), write_enable(3G).

NAME

read_choice_event – read choice event from top of event queue

SYNOPSIS

C Syntax:

```
int read_choice_event(queue_des,choice_fildes,ordinal,value,
        status,message_link);
int queue_des;
int *choice_fildes,*ordinal,*value,*status,*message_link;
```

FORTRAN77 Syntax:

```
integer*4 function read_choice_event(queue_des,
        choice_fildes,ordinal,value,status,message_link)
integer*4 queue_des,choice_fildes,ordinal
integer*4 value,status,message_link
```

Pascal Syntax:

```
function read_choice_event(queue_des:integer;
var choice_fildes,ordinal,value,
        status,message_link:integer):integer;
```

DESCRIPTION

Input Parameters

queue_des File descriptor of an opened graphics device or −1 (minus one). If it is a file descriptor, the first event for that device is read. If −1 (minus one), the next event from any input device is read.

Output Parameters

choice_fildes File descriptor of the device returning the event.

ordinal Device number returning the event. To determine maximum device number, use *inquire_sizes*.

value Value returned from a choice device.

status Set to one of the following defined values:

EMPTY_NO_OVERFLOW
NOT_EMPTY_NO_OVERFLOW
EMPTY_OVERFLOW
NOT_EMPTY_OVERFLOW.

message_link Set to one of the defined values:

SINGLE_EVENT
SIMULTANEOUS_EVENT_FOLLOWS

Discussion

If there is a choice event on top of the event queue, *read_choice_event* returns the top event. If the queue is empty or the top event is not a choice event, the call returns a non-zero function value and the output parameters are undefined.

If there are more simultaneous events in the queue, *message_link* is set to SIMULTANEOUS_EVENT_FOLLOWS. Otherwise, *message_link* is set to SINGLE_EVENT.

Status is set to the state of the queue after the current event report has been removed from the queue (either of the two EMPTY states indicates that there are no more events currently in the queue). The event queue can hold up to 100 events. If the queue is full, any new events are lost. If events are lost, the next **status** returned indicates OVERFLOW and the overflow flag is cleared.

A single event queue services all devices and events are entered into the queue in the cronological order in which they occur. The order in which simultaneous events are queued is implementation-dependent, but each simultaneous event entry in the queue, except for the last entry, indicates SIMULTANEOUS_EVENT_FOLLOWS.

SEE ALSO

read_locator_event(3G).

NAME

read_locator_event – read locator event from top of event queue

SYNOPSIS

C Syntax:

```
int read_locator_event(queue_des,locator_fildes,ordinal,
        x,y,z,status,message_link);
int queue_des,*locator_fildes,*ordinal,*status,*message_link;
float *x,*y,*z;
```

FORTRAN77 Syntax:

```
integer*4 function read_locator_event(queue_des,
        locator_fildes,ordinal,x,y,z,status,message_link)
integer*4 queue_des,locator_fildes
integer*4 ordinal,status,message_link
real x,y,z
```

Pascal Syntax:

```
function read_locator_event(queue_des:integer;
var locator_fildes,ordinal:integer;var x,y,z:real;
var status,message_link:integer):integer;
```

DESCRIPTION

Input Parameters

queue_des File descriptor of an opened graphics device or −1. If **queue_des** is the file descriptor of an opened graphics device, the first event for that device is read. If **queue_des** is −1, the top event from any input device is read.

Output Parameters

locator_fildes File descriptor of the device returning the event.

ordinal Device number returning the event. To determine maximum device number, use *inquire_sizes*.

x,y,z Point returned from a locator.

status Set to one of the following defined values:

EMPTY_NO_OVERFLOW
NOT_EMPTY_NO_OVERFLOW
EMPTY_OVERFLOW
NOT_EMPTY_OVERFLOW

Message_link Set to one of the following defined values:

SINGLE_EVENT
SIMULTANEOUS_EVENT_FOLLOWS

Discussion

If there is a locator event on top of the event queue, *read_locator_event* returns the top event. If the queue is empty or the top event is not a locator event, the call returns a non-zero function value and the output parameters are undefined.

If there are more simultaneous events in the queue, **message_link** is set to SIMULTANEOUS_EVENT_FOLLOWS. Otherwise, **message_link** is set to SINGLE_EVENT.

Status is set to the state of the queue after the current event report has been removed from the queue (either of the two EMPTY states indicates that there are no more events currently in the queue). The event queue can hold up to 100 events. If the queue is full, any new events are lost. If events are lost, the next **status** returned indicates OVERFLOW and the overflow flag is cleared.

A single event queue services all devices and events are entered into the queue in the cronological order in which they occur. The order in which simultaneous events are queued is implementation-dependent, but each simultaneous event entry in the queue, except for the last entry, indicates SIMULTANEOUS_EVENT_FOLLOWS.

SEE ALSO

await_event(3G), enable_events(3G), disable_events(3G).

NAME

dcrectangle, intrectangle, rectangle – define rectangular region to be filled and/or edged

SYNOPSIS

C Syntax:

```
void dcrectangle(fildes,dcx1,dcy1,dcx2,dcy2);
int fildes,dcx1,dcy1,dcx2,dcy2;

void intrectangle(fildes,x1,y1,x2,y2);
int fildes,x1,y1,x2,y2;

void rectangle(fildes,x1,y1,x2,y2);
int fildes;
float x1,y1,x2,y2;
```

FORTRAN77 Syntax:

```
subroutine dcrectangle(fildes,dcx1,dcy1,dcx2,dcy2)
integer*4 fildes,dcx1,dcy1,dcx2,dcy2

subroutine intrectangle(fildes,x1,y1,x2,y2)
integer*4 fildes,x1,y1,x2,y2

subroutine rectangle(fildes,x1,y1,x2,y2)
integer*4 fildes
real x1,y1,x2,y2
```

Pascal Syntax:

```
procedure dcrectangle(fildes,dcx1,dcy1,dcx2,dcy2:integer);

procedure intrectangle(fildes,x1,y1,x2,y2:integer);

procedure rectangle(fildes:integer;x1,y1,x2,y2:real);
```

DESCRIPTION

Input Parameters

fildes	Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.
x1,y1	First corner of the rectangle in world coordinates.
x2,y2	Second corner of the rectangle in world coordinates.
dcx1,dcy1	First corner of the rectangle in device coordinates.
dcx2,dcy2	Second corner of the rectangle in device coordinates.

Discussion

A boundary of a rectangular region is defined by defining diagonally opposite corners of the rectangle. If the transform mode has been set to 3d, the **z** value used is that of the current position.

The rectangle is filled and/or outlined according to the current interior style, using the current fill color and perimeter attributes. As with all output primitives, *rectangle* is affected by the current drawing mode and write enable.

No clipping or transformations are performed on device coordinate procedures.

Device coordinate current pen position and world device coordinate current pen position are not related. If you are switching between the two coordinate systems, always begin the use of either system with a *move* to a known location before performing any other operations.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer

operations, floating point operations and common operations see the *starbase.3g* manual page.

SEE ALSO

drawing_mode(3G), fill_color(3G), interior_style(3G), perimeter_color(3G), perimeter_type(3G), perimeter_repeat_length(3G), polygon(3G), write_enable(3G).

NAME

replace_matrix2d, replace_matrix3d, intreplace_matrix2d – replace current transformation matrix with a specified matrix

SYNOPSIS

C Syntax:

```
void intreplace_matrix2d(fildes,xform2,radix,raw);
int fildes,xform2[3][2],radix,raw;

void replace_matrix2d(fildes,xform2);
int fildes;
float xform2[3][2];

void replace_matrix3d(fildes,xform3);
int fildes;
float xform3[4][4];
```

FORTRAN77 Syntax:

```
See Language Dependencies Below

subroutine intreplace_matrix2d(fildes,xform2,radix,raw)
integer*4 fildes,xform2(2,3),radix,raw

subroutine replace_matrix2d(fildes,xform2)
integer*4 fildes
real xform2(2,3)

subroutine replace_matrix3d(fildes,xform3)
integer*4 fildes
real xform3(4,4)
```

Pascal Syntax:

```
type
int2d_xform=array[1..3][1..2] of integer;
two_d_xform=array[1..3][1..2] of real;
three_d_xform=array[1..4][1..4] of real;

procedure intreplace_matrix2d(fildes:integer;var xform2:int2d_xform;
                radix,raw:integer);

procedure replace_matrix2d(fildes:integer;xform2:two_d_xform);

procedure replace_matrix3d(fildes:integer;var xform3:three_d_xform);
```

DESCRIPTION

Input Parameter

fildes	Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.
xform2	A 3x2 (2-dimensional) matrix.
xform3	A 4x4 (3-dimensional) matrix.
radix	is the radix factor for the 3x2 integer matrix.
raw	If set TRUE(1), integer matrices will be in internal (raw) format (i.e. translation values are not scaled). If set to FALSE(0), all integer matrix values are scaled.

Discussion

The current transformation matrix (top of matrix stack) is replaced by the specified matrix, **xform2** or **xform3**. The matrix is concatenated with the current viewing transformation matrix before the replacement is made. If the graphics device has not been opened in MODEL_XFORM

mode, the matrix is concatenated with the current viewing transformation matrix before the replacement is made. The current transformation matrix is used to transform subsequent output primitives.

The vdc-to-device units-transformation matrix cannot be replaced using *replace_matrix2d*, *intreplace_matrix2d*, or *replace_matrix3d*. It can be changed using *intvdc_extent*, *vdc_extent*, *vdc_justification*, *set_p1_p2*, or *mapping_mode*. Likewise, the viewing transformation matrix cannot be replaced using these functions. It can be changed using *intview_matrix2d*, *intview_port*, *intview_window*, *view_camera*, *view_port*, *view_matrix*, *view_volume*, or *view_window*. If there are no matrices in the matrix stack, *replace_matrix* generates a warning.

The coordinate systems used by Starbase may be conceptually defined as follows:

1. It is assumed that user points are defined in modelling coordinates. These points are transformed into world coordinates using a matrix called the *modelling transformation matrix* (provided a matrix has been pushed on the stack). World coordinates are used to perform any rendering calculations needed. Use *replace_matrix* to place modelling transformations on the matrix stack.
2. World coordinates are then transformed to device coordinates by the *viewing transformation matrix*, which is usually the concatenation of user-defined viewing transformations and the vdc-to-device transformation matrix. When a device is opened the *viewing transformation matrix* consists of the vdc-to-device coordinate transformation matrix. Viewing functions can be used to define further viewing transformations such as perspective. When a device is opened the *viewing transformation matrix* is simply the vdc-to-device coordinated transformation matrix. *Intview_window*, *intview_port*, *intview_matrix2d*, *view_window*, *view_port*, *view_matrix*, *view_camera*, and *view_volume* can be used to define further viewing transformations.

If a graphics device has been opened in MODEL_XFORM mode, special rendering calculations such as shading may be needed after the modelling-to-world coordinate transformation. For this reason, the modelling transformation cannot be combined with the viewing transformation. Thus, matrices pushed on the stack are left as is and all transformations from modelling coordinates to device coordinates occur in two steps: modelling to world coordinates followed by world-to-device coordinates.

If a graphics device is not in MODEL_XFORM mode then the modelling and viewing transformations may be combined; thus the current viewing transformation matrix is post-concatenated to modelling matrices placed on the matrix stack and subsequent output primitives are transformed by the top matrix on the matrix stack only.

Starbase maintains all internal floating point matrices in 3-dimensional (4x4) form, so *replace_matrix2d* must expand the matrix **xform2**.

Perspective transformations (as well as any other non-linear or viewing transformation) should not appear in modelling matrices since this will distort surface normals, invalidating lighting calculations; rather, they should appear in the viewing transformation which is set with *view_matrix*. When performing perspective transformations, a perspective model where the eye is at origin of perspective space is recommended. Any other model can easily be modified by a single translation step.

Integer transformation matrices are scaled to allow a fractional portion for rotating objects. The radix factor indicates the number of bits to the right of the decimal point. Legal limits are 0 to 30. Once a coordinate has been transformed, it is divided by **2**radix** to return to an integer value.

When using raw mode with an integer matrix, positions (3,1) and (3,2) have an implied radix factor of 0 (no scaling). This allows large translation ranges along with accurate rotations.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode Floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of Integer operations, Floating point operations and common operations see the *starbase.3g* manual page.

LANGUAGE DEPENDENCIES

FORTRAN77:

A transposition of array rows and columns is required due to the manner in which FORTRAN77 stores arrays.

DEFAULTS

After *gopen*, the current transformation matrix is the vdc-to-device units transformation matrix.

SEE ALSO

concat_matrix(3G), concat_transformation(3G), gopen(3G), mapping_mode(3G), push_matrix(3G), pop_matrix(3G), push_vdc_matrix(3G), set_p1_p2(3G), vdc_extent(3G), view_matrix(3G), vdc_justification(3G).

NAME

request_choice – wait for input device to be triggered then return measured value

SYNOPSIS

C Syntax:

```
void request_choice(fildes,ordinal,timeout,valid,value);
int fildes,ordinal,*valid,*value;
float timeout;
```

FORTRAN77 Syntax:

```
subroutine request_choice(fildes,ordinal,timeout,
      valid,value)
integer*4 fildes,ordinal,valid,value
real timeout
```

Pascal Syntax:

```
procedure request_choice(fildes,ordinal:integer;
      timeout:real;var valid,value:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

ordinal Logical device number ranging from 1 to number of devices of this class.

timeout Time, in seconds, to wait for a trigger.

Output Parameters

valid TRUE (1 in Pascal) if data is valid.

FALSE (0 in Pascal) if data is invalid (timeout period exceeded or device not initialized). If FALSE, **value** is undefined.

value Set to the current measure value if **valid** is set to TRUE (1 in Pascal)

Discussion

Request_choice returns the measure value of the device specified by **fildes** and **ordinal** when a trigger occurs, or when when the time period specified by **timeout** is exceeded. If the request was not previously posted with *initiate_request* and no request data is ready for this device, it is posted by this call. If a request was previously posted but not yet satisfied, the timeout runs from the time this call was made.

A request can be performed on an input device while it is enabled for event generation. The mode of (events enabled/events disabled) is saved by the device and restored after the request has been satisfied. The request data does not go into the event queue, but into a separate virtual register.

Use *inquire_request_status* to determine whether a previously posted request is ready.

To discard a request that was previously posted and satisfied, but which has not been read, use *request_choice* with a zero **timeout** and discard the result.

SEE ALSO

initiate_request(3G), inquire_request_status(3G), request_locator(3G).

NAME

intrequest_locator2d, request_locator – wait for input device to be triggered then return measured value

SYNOPSIS

C Syntax:

```
void request_locator(fildes,ordinal,timeout,valid,x,y,z);
int fildes,ordinal,*valid;
float timeout,*x,*y,*z;

void intrequest_locator2d(fildes,ordinal,timeout,valid,x,y);
int fildes,ordinal,*valid,*x,*y;
float timeout;
```

FORTRAN77 Syntax:

```
subroutine request_locator(fildes,ordinal,timeout,valid,x,y,z)
integer*4 fildes,ordinal,valid
real timeout,x,y,z

subroutine intrequest_locator2d(fildes,ordinal,timeout,valid,x,y)
integer*4 fildes,ordinal,valid,x,y
real timeout
```

Pascal Syntax:

```
procedure request_locator(fildes,ordinal:integer;timeout:real;
        var valid:integer;var x,y,z:real);

procedure intrequest_locator2d(fildes,ordinal:integer;timeout:real;
        var valid,x,y:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

ordinal Logical device number ranging from 1 to number of devices of this class.

timeout Time, in seconds, to wait for a trigger.

Output Parameters

valid TRUE (1 in Pascal) if data is valid.

FALSE (0 in Pascal) if data is invalid (timeout period exceeded or device not initialized). If FALSE, **x,y,z** is undefined.

x,y,z Set to x, y, and z virtual device coordinate values of the point returned from a locator device.

Discussion

Request_locator returns the measured value of the device specified by **fildes** and **ordinal** when a trigger occurs, or when when the time period specified by **timeout** is exceeded. If the request was not previously posted with *initiate_request* and no request data is ready for this device, it is posted by this call. If a request was previously posted but not yet satisfied, the timeout runs from the time this call was made.

A request can be performed on an input device while it is enabled for event generation. The mode of (events enabled/events disabled) is saved by the device and restored after the request has been satisfied. The request data does not go into the event queue, but into a separate virtual register.

Use *inquire_request_status* to determine whether a previously posted request is ready.

To discard a request that was previously posted and satisfied, but which has not been read, use *request_choice*, *request_locator*, or *intrequest_locator2d* with a zero **timeout** and discard the result.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

SEE ALSO

initiate_request(3G), inquire_request_status(3G).

NAME

rgb_to_index – find the index of the closest color in the color map

SYNOPSIS

C Syntax:

```
int rgb_to_index(fildes,red,green,blue)
int fildes;
float red,green,blue;
```

FORTRAN77 Syntax:

```
integer*4 function rgb_to_index(fildes,red,green,blue)
integer*4 fildes
real red,green,blue
```

Pascal Syntax:

```
function rgb_to_index(fildes:integer;red,green,blue:real):integer;
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

red,green,blue Color values (in the range of 0.0 to 1.0) of the desired color.

Discussion

This call searches the color table for the closest color to (red,green,blue) and returns its index. The distance between colors is defined by the Euclidean metric--see the "Color" Chapter of the *Starbase Graphics Techniques; HP-UX Concepts and Tutorials* manual for details.

This routine cannot be used in CMAP_FULL color mapping mode.

If the color map or double buffer mode is modified, previously returned indices may be invalidated.

SEE ALSO

define_color_table(3G), inquire_color_table(3G), shade_mode(3G)

NAME

sample_choice – return current choice value

SYNOPSIS

C Syntax:

```
void sample_choice(fildes,ordinal,valid,value);
int fildes,ordinal,*valid,*value;
```

FORTRAN77 Syntax:

```
subroutine sample_choice(fildes,ordinal,
      valid,value)
integer*4 fildes,ordinal,valid,value
```

Pascal Syntax:

```
procedure sample_choice(fildes,ordinal:integer;
var valid,value:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

ordinal Logical device number ranging from 1 to number of devices in this class.

Output Parameters

valid TRUE (1) if a valid choice was returned, FALSE (0) otherwise.

value Current choice value.

Discussion

Sample_choice returns the current value of the choice device specified by **fildes** and **ordinal**. This is the equivalent of a non-blocking read.

SEE ALSO

request_choice(3G).

NAME

intsample_locator2d, sample_locator – return current locator value

SYNOPSIS

C Syntax:

```
void intsample_locator2d(fildes,ordinal,valid,x,y);
int fildes,ordinal,*valid,*x,*y;

void sample_locator(fildes,ordinal,valid,x,y,z);
int fildes,ordinal,*valid;
float *x,*y,*z;
```

FORTRAN77 Syntax:

```
subroutine intsample_locator2d(fildes,ordinal,valid,x,y,z)
integer*4 fildes,ordinal,valid,x,y

subroutine sample_locator(fildes,ordinal,valid,x,y,z)
integer*4 fildes,ordinal,valid
real x,y,z
```

Pascal Syntax:

```
procedure intsample_locator2d(fildes,ordinal:integer;var valid:integer;
        var x,y:integer);

procedure sample_locator(fildes,ordinal:integer;
var valid:integer;var x,y,z:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

ordinal Logical device number ranging from 1 to number of devices in this class.

Output Parameters

valid TRUE (1) if a valid point was returned,
FALSE (0) otherwise.

x,y,z x, y, and z virtual device coordinate values of a point returned from the locator device.

Discussion

Sample_locator returns the current value of the locator device specified by **fildes** and **ordinal**. This is the equivalent of a non-blocking read.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not abailable for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations, and common operations, se the *starbase.3g* manual page.

SEE ALSO

request_locator(3G).

NAME

set_locator – set locator value

SYNOPSIS

C Syntax:

```
void set_locator(fildes,ordinal,x,y,z);
int fildes,ordinal;
float x,y,z;
```

FORTRAN77 Syntax:

```
subroutine set_locator(fildes,ordinal,x,y,z)
integer*4 fildes,ordinal
real x,y,z
```

Pascal Syntax:

```
procedure set_locator(fildes,ordinal:integer;x,y,z:real);
```

DESCRIPTION

INPUT PARAMETER

fildes Integer file descriptor returned by *gopen* when the I/O path to a graphics device is opened.

ordinal

Logical device number ranging from 1 to the number of devices in the locator class.

x,y,z Define the new x, y and z virtual device coordinate values of the locator device.

Discussion

Set_locator sets the value of the locator device specified by **fildes** and **ordinal**. For a relative-position device, such as a mouse, this call sets the **x,y,z** coordinates to the current device location. For an absolute device such as a tablet, the location is also set, but it is then reset by the device to the current location of the tablet's pen.

NAME

set_p1_p2 – set physical device limits

SYNOPSIS

C Syntax:

```
void set_p1_p2(fildes,units,p1_x,p1_y,p1_z,p2_x,p2_y,p2_z);
int fildes,units;
float p1_x,p1_y,p1_z,p2_x,p2_y,p2_z;
```

FORTRAN77 Syntax:

```
subroutine set_p1_p2(fildes,units,p1_x,p1_y,p1_z,
p2_x,p2_y,p2_z)
integer*4 fildes,units
real p1_x,p1_y,p2_z,p2_x,p2_y,p2_z
```

Pascal Syntax:

```
procedure set_p1_p2(fildes,units:integer;p1_x,p1_y,p1_z,
p2_x,p2_y,p2_z:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

units If set to FRACTIONAL, parameters for P1 and P2 represent decimal fractions of the physical device limits where (0.0,0.0,0.0) represents the lower-left-front corner; (1.0,1.0,1.0) represents the upper-right-rear corner of the physical display surface. All values of P1 and P2 must range from 0.0 to 1.0, denoting the fraction of the total surface in x or y. Units are anisotropic except in the rare case of a square device.

If set to METRIC, area specification is represented in millimetres. To determine current P1 and P2 in millimetres, use starbase *inquire_sizes*(3G).

p1_x,p1_y,p1_z Specify P1 (lower-left-front corner) of the region of interest.

p2_x,p2_y,p2_z Specify P2 (upper-right-rear corner) of the region of interest.

Discussion

P1 can be in any position relative to P2, possibly causing mirroring in either or both axes.

This procedure updates the current vdc-to-device units transformation matrix. If no matrices have been placed on the matrix stack and no viewing transformations have been defined, the new vdc matrix becomes the current viewing transformation. Otherwise no other transformation matrices are affected.

P1 and P2 also define the portion of the display surface to be rendered using hidden surface removal if it is enabled. *Set_p1_p2* cannot be called while hidden surface removal is enabled because *set_p1_p2* might alter the number of passes required.

Set P1 and P2 before setting related values because changes to P1 and P2 may not affect previously set values such as text size or tracking relationships.

DEFAULTS

Units is FRACTIONAL. P1 = (0,0,0) and P2 = (1,1,1).

SEE ALSO

hidden_surface(3G). inquire_sizes(3G), mapping_mode(3G), vdc_extent(3G), vdc_justification(3G),

NAME

set_signals – disable or enable signal function of specified device

SYNOPSIS

C Syntax:

```
void set_signals(fildes,signal);
int fildes,signal;
```

FORTRAN77 Syntax:

```
subroutine set_signals(fildes,signal)
integer*4 fildes,signal
```

Pascal Syntax:

```
procedure set_signals(fildes,signal:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

signal Signal number to send or zero (0) to disable signals.

Discussion

Set_signals disables or sets the **signal** function of the device identified by **fildes**. A non-zero **signal** value defines the signal to be sent if an event is generated. If **signal** is zero, no signals are sent.

DEFAULTS

signal = 0: No signal to be sent.

SEE ALSO

signal(2).

NAME

shade_mode – enable/disable light source polygon shading.

SYNOPSIS

C Syntax:

```
void shade_mode(fildes,mode,shading);
int fildes,mode,shading;
```

FORTRAN77 Syntax:

```
shade_mode(fildes,mode,shading)
integer*4 fildes,mode,shading
```

Pascal Syntax:

```
procedure shade_mode(fildes,mode,shading:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *open* when an I/O path to a graphics device is opened.

mode Specifies Starbase color map setup: CMAP_NORMAL, CMAP_MONOTONIC, and CMAP_FULL. Can be ORed with INIT to initialize color map.

shading
If TRUE (1), filled polygons are processed through the light source equations.

Discussion

The *mode* parameter specifies how the color map is interpreted:

CMAP_NORMAL
This is the default mode. The color table is searched for the color closest to the one requested for procedures such as *fill_color* and *line_color*. CMAP_NORMAL cannot be used if **shading** or depth cueing is on, or color coordinates are being used for output primitives. If INIT is ORed with CMAP_NORMAL, the color map is initialized to its default state. In this mode, a 24-plane graphics device behaves as a standard 8-plane device.

CMAP_MONOTONIC
This mode assumes a monotonically increasing intensity between the color map indices set by *shade_range*. All color values are converted to intensities by the equation:

I=0.30(red)+0.59(green)+0.11(blue)

This intensity value is then mapped to an index using the **min** and **max** specified by *shade_range*. If INIT is ORed with CMAP_MONOTONIC, the color map is initialized to a linear grey scale between the lowest index and the highest index. A 24-plane graphics device behaves like a standard 8-plane device.

CMAP_FULL In this mode, color mapping is assumed as follows:

(8-plane) or (16 plane double-buffered) graphics device:
3 bits red, 3 bits green, and 2 bits blue.

(4-plane) or (8 plane double-buffered) graphics device:
1 bit red, 2 bits green, and 1 bit blue.

(3-plane) or (6 plane double-buffered) graphics device:

1 bit red, 1 bit green, and 1 bit blue.

(6-plane) graphics device:
2 bits red, 2 bits green, and 2 bits blue.

(24-plane) graphics device:
8 bits red, 8 bits green, and 8 bits blue or
4 bits red, 4 bits green, and 4 bits blue if double-buffered.

CMAP_FULL can be specfied only if the graphics device has 3 or more planes. CMAP_FULL requires at least 3 planes per buffer when double-buffering is used. If INIT is ORed with CMAP_FULL, the color map is initialized according to the above assumptions.

(Note: the 3-D accelerated devices require at least 16 planes if double buffered in CMAP_FULL mode.

When shading is FALSE (0), light sources are not applied to polygons. When shading is TRUE (1), the light source equations, as defined by *surface_model* are used to determine the polygon colors. Note that if all the lights are disabled by *light_switch*, polygons will not be seen. The *vertex_format* procedure is used to obtain various shading effects such as constant (faceted) shading and RGB interpolation (smooth or Gouraud) shading.

Current colors are recalculated using the current color map mode.

shade_mode may change current bank usage (see *bank_switch*). Since unused banks are sometimes used for the zbuffer when hidden surface removal is enabled, changing the color map mode when hidden surface removal is enabled is not permitted. For example, if the color map mode is CMAP_NORMAL when hidden surface removal is turned on, no more than two banks will be in use. If additional banks are available in the device, they may be allocated for zbuffer usage. This means that if the color map mode were later changed to CMAP_FULL, up to three banks could be used for image space, resulting in a collision if the third bank was being used for the zbuffer. Such a change could also increase the number of passes needed to fully render an image, so simply recalculating the zbuffer allocation would not produce a satisfactory solution. For this reason, the color map mode should be set before hidden surface removal is enabled. If the color map mode needs to be changed, hidden surface removal should be disabled before the change is made.

Performance Note: Color calculations for *fill_color, line_color, perimeter_color, background_color, text_color,* and *marker_color are much* faster for the CMAP_FULL and CMAP_MONOTONIC modes than the CMAP_NORMAL mode. This is because the colors can be calculated as opposed to searching the color map for the closest colors.

See the *Starbase Programming with X11* chapter on display control for more information about using this routine with a window system.

DEFAULTS

mode = CMAP_NORMAL,
shading = FALSE (0).

SEE ALSO

inquire_display_mode(3G), light_source(3G), light_switch(3G), shade_range(3G), surface_model(3G), vertex_format(3G), *Starbase Graphics Techniques, Starbase Programming with X11.*

NAME

shade_range – set intensity-to-frame-buffer-index mapping

SYNOPSIS

C Syntax:

```
void shade_range(fildes,min,max);
int fildes,min,max;
```

FORTRAN77 Syntax:

```
subroutine shade_range(fildes,min,max)
integer*4 fildes,min,max
```

Pascal Syntax:

```
procedure shade_range(fildes,min,max:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *open* when an I/O path to a graphics device is opened.

min Frame buffer index that intensity 0.0 maps to.

max Frame buffer index that intensity 1.0 maps to.

Discussion

When CMAP_MONOTONIC is specified in *shade_mode*, subsequent red, green, blue values are converted to an intensity, then mapped into the frame buffer using **min** and **max**. The **min** value must be less than the **max** value. It is left to the user to make sure the color map is set up in a monotonically increasing intensity between the two values. See *shade_mode*(3G) for the formula used to convert **red,green,blue** values to intensity.

DEFAULTS

min = 0; **max** is set to highest available index value.

SEE ALSO

shade_mode (3G), *Starbase Graphics Techniques.*

NAME

spline_curve2d, spline_curve3d, spline_surface – draw a space curve or surface.

SYNOPSIS

C Syntax:

```
void spline_curve2d (fildes,plist,numpts,order,rational);
int fildes, numpts, order, rational;
float plist[];

void spline_curve3d (fildes,plist,numpts,order,rational);
int fildes, numpts, order, rational;
float plist[];

void spline_surface(fildes,plist,numpts_u,numpts_v,order_u,order_v,rational);
int fildes, numpts_u, numpts_v, order_u, order_v,
float plist[];
```

FORTRAN77 Syntax:

```
subroutine spline_curve2d(fildes,plist,numpts,order,rational)
integer*4 fildes, numpts, order, rational
real plist (numpts*3)

subroutine spline_curve3d(fildes,plist,numpts,order,rational)
integer*4 fildes, numpts, order, rational
real plist (numpts*3)

subroutine spline_surface (fildes,plist,numpts_u,numpts_v,
        order_u,order_v,rational)
integer*4 fildes, numpts_u, numpts_v, order_u, order_v,
real plist (numpts_u*numpts_v*3)
```

Pascal Syntax:

```
procedure spline_curve2d(fildes:integer;
        var plist array [lo..hi:integer] of real;
        numpts, order, rational:integer);

procedure spline_curve3d(fildes:integer;
        var plist array [lo..hi:integer] of real;
        numpts, order, rational:integer);

procedure spline_surface(fildes:integer;
        var plist array[lo..hi:integer] of real;
        numpts_u, numpts_v, order_u, order_v, rational:integer);
```

DESCRIPTION

Input Parameters

fildes	Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.
numpts	Number of points in the plist array
numpts_u,numpts_v	Number of points along each axis in the plist array
plist	Array of spline control points
order, order_u, order_v	space curve or surface order: LINEAR = 2, QUADRATIC = 3, CUBIC = 4, QUARTIC = 5, QUINTIC = 6.
rational	Specifies whether the spline is RATIONAL (1) or NONRATIONAL (0)

Discussion

spline_curve draws a spline curve of the specified order, using the points in the plist array as control points and the current interpolation matrices specified by the *u* knot vector. The curve is drawn using the current polyline attributes and the u_exterior *curve_resolution*.

spline_surface draws a spline surface of the specified orders, using the points in the plist array as control points and the current interpolation matrices specified by the *u* and *v* knot vectors. Use *curve_resolution* to set the size of polylines or polygons that make up the curve or surface.

The data in the *plist* array is two dimensional coordinate data (x,y), or (x,y,h) for **spline_curve2d**, three dimensional (x,y,z) or (x,y,z,h) for **spline_curve3d**, and three dimensional (x,y,z) or (x,y,z,h) for **spline_surface** where h is the rational coordinate. The **spline_surface** data is ordered first along the u-axis and then along the v-axis; that is, the individual rows of the **plist** array are assumed to contain the data to be interpolated for a constant v value.

The current *vertex_format* affects spline surface generation. If **coord** = 3, **use** = 3, and **rgb** = FALSE (normals per vertex), spline surface generation generates normals per vertex and performs smooth (Gouraud) shading. Constant (faceted) shading is used for spline surfaces in all other cases. The direction of these normals is effected by the clockwise parameter of vertex_format. If counter clockwise polygons are specified then the normals are calculated as the u-derivative cross the v-derivative. If clockwise polygons are specified then the normals are calculated as the v-derivative cross the u-derivative. Spline surfaces also use the current **interior_style**, so **interior_style** must be set to INT_SOLID for any shading to occur.

Surfaces that degenerate to a line or point are illegal if normals per vertex are specified in *vertex_format*.

SEE ALSO

curve_resolution(3G), knot_vectors(3G), trimming_curve(3G), vertex_format(3G), *Starbase Graphics Techniques.*

NAME

starbase – Starbase Graphics Library description

DESCRIPTION

The Starbase Graphics Library consists of procedures used to generate graphics text and pictures. This library is based on the emerging American National Standards Institute's Computer Graphics Interface standard (ANSI CGI). The Starbase Graphics Library provides a high-performance interface to HP graphics hardware and other selected graphics peripherals. These library routines can be used in C, Pascal and Fortran77 programs. In addition to the standard ANSI-CGI functions, Starbase provides asynchronous input functions, 3-dimensional transformations, and access to many device-dependent hardware features.

LOADING STARBASE PROGRAMS

It is extremely important that link files be specified in the following order:

driver(s),
libsb1.a,
libsb2.a .

Some drivers may require that additional libraries be linked with them. For instance, using *libddhpgl.a* or *libCADplt.a* requires that *libdvio.a* be linked as well. Also, some functions in the Starbase library require that the math library be linked in. Functions that require the math library are *view_camera, ellipse, arc, intarc, intcircle, inttext2d, text, intline_width, line_width, dccircle,* and the *shading* functions. Specific requirements are described in the *Starbase Device Drivers Library Manual*.

For example: to compile and link the C program *test.c* for running on a terminal, hp98720 or hpgl plotter, use:

cc test.c -lddhpterm -ldd98720 -lhpgl -lsb1 -lsb2 -ldvio -lm -o test -O

LOADING STARBASE HP WINDOWS/9000 PROGRAMS

Series 300 only

To load a Starbase program that may be directed to a window, the window library and possibly the byte driver must be included. The byte driver needs to be included for programs that draw to obscured graphics windows which are retained. Again, the order is important:

driver(s), [byte_driver], window_library, libsb1.a, libsb2.a

For example:

cc test.c -ldd98720 -lddbyte -lwindow -lsb1 -lsb2 -lm -o test -O

For some monochrome devices, the bit driver is an alternative to the byte driver. This is because the bit driver requires one eighth the memory for a single-plane retained raster.

LOADING STARBASE X11 WINDOW SYSTEM PROGRAMS

To load a Starbase program that may be directed to an X11 window, you need to include the X11 window libraries, and possibly the byte driver. The byte driver needs to be included for programs that draw to obscured graphics windows that have backing store. Again, the order is important:

driver(s), [byte_driver], window_library, libsb1.a, libsb2.a, libXhp11.a, libX11.a

For example:

cc test.c -ldd98720 -lddbyte -lXwindow -lsb1 -lsb2 -lXhp11 -lX11 -lm test -o

For some monochrome devices, the bit driver is used instead of the byte driver because the bit driver requires one-eighth the memory for a single-plane retained raster.

INCLUDE FILE USAGE

The following templates show how include files should be used with the various language

bindings:

C include Template

```
#include <starbase.c.h>
func()
{
        /* function body */
}
main()
{
        /* program body */
}
```

FORTRAN include Template

```
        include '/usr/include/starbase.f1.h'
        program main
C       PARAMETER statements
        character NULL
        parameter(NULL = char(0))
        include '/usr/include/starbase.f2.h'
C       VARIABLE declarations
C       program body and/or subroutines and functions
        end
```

Pascal include Template

```
program main(input,output);
$include '/usr/include/starbase.p1.h'$
{ CONSTANTS }
{ TYPES }
{ VARIABLES }
$include '/usr/include/starbase.p2.h'$
{ PROCEDURES AND FUNCTIONS }
begin
{program body}
end.
```

FILES

The following files make up the Starbase Graphics Library. Each is shown with a description of its function. These files are located in the **/usr/lib** directory.

libsb1.a	Starbase Main Library
libsb2.a	Starbase Stub Entry Points

(For series 300 only.)

libddbit.a	HP 9000/Windows and X11 memory file (bit/pixel)
libddbyte.a	HP 9000/Windows and X11 memory file (byte/pixel)
libwindow.a	HP 9000/Windows Library
libXwindow.a	X11 Starbase support Library

and device drivers of the form libdd*.a

The following files are found in the **/usr/include** directory and should be used with the appropriate programming language.

starbase.c.h	C program header file
starbase.f1.h	Fortran77 program header file
starbase.f2.h	Fortran77 program header file
starbase.p1.h	Pascal program header file
starbase.p2.h	Pascal program header file

The files residing in the **/usr/lib/starbase** directory with the form sb_daemon_*.* are the Starbase input daemons. The *.* is the version number of the daemon.

The files residing in the **/usr/lib/starbase/hp*** directory (where * is the hardware type) named te_mcode are the micro code for Starbase. Those in the same directories named te_data are the micro code data files.

The Starbase stroke font files reside in **/usr/lib/starbase/stroke/*** where * is the type of font used.

The error catalog files reside in either **/usr/lib/nls/C/n-computer/sb.cat** (pre-7.0 release) or in **/usr/lib/nls/C/sb.cat** (7.0 release and after).

The file **/usr/lib/starbase/defaults** contains information about what fonts are loaded during a *gopen,* and what the character switching mode is after a *gopen.* Entries in this file may be modified, and will effect all *gopens* done on the workstation. The comments in the file describe what text or font attribute is affected.

SIGNALS

It is not recommended to use Starbase calls within a signal handler. There is state information within a procedure that can be modified or lost if a second Starbase procedure is called while the first procedure is processing that information. If it is necessary to use Starbase calls, keep track of the current active Starbase file descriptor, and use a different file descriptor to the same device. Make sure this second file descriptor is not used outside the signal handler. To reset the device, at the end of the signal handler make a call to the current active device using a Starbase procedure that does not actually access the device, such as *inquire_color_table* with a count of 0.

STARBASE COORDINATE SYSTEMS

There are three Starbase coordinate systems: Floating point Virtual Device Coordinates, Integer Virtual Device Coordinates and Device Coordinates. Floating point operations are either 2d or 3d, and provide functions for light sources, shading, and hidden surface removal. Floating point operations are available by default, or by using the FLOAT_XFORM gopen mode. Integer operations provide only 2d transformations, and are available only when using the INT_XFORM gopen mode. When using INT_XFORM mode, Floating point operations are not available. Device coordinate operations are always available. Although the majority of Starbase procedures are used with only one coordinate system, a few of them are used with two and some with all three. A list of Starbase procedures and which system they are used with appears below.

Floating point only:

append_text
arc
backface_control
bitmap_print
bitmap_to_file
block_move
block_read
block_write
character_height
character_width
clip_depth
clip_rectangle
concat_matrix
concat_transformation2d
concat_transformation3d
dc_to_vdc
default_knots
define_trimming_curve
depth_cue
depth_indicator
draw2d
draw3d
echo_type
echo_update
ellipse
file_to_bitmap
hatch_spacing
hidden_surface
inquire_current_position2d
inquire_current_position3d
inq_pick_depth
inq_pick_window
inquire_text_extent
light_ambient
light_model
light_source
light_switch
line_repeat_length
marker_orientation
marker_size
move2d
move3d
partial_arc
partial_ellipse
partial_polygon2d
partial_polygon3d
perimeter_repeat_length
polygon2d
polygon3d
polyline2d
polyline3d
polymarker2d
polymarker3d
pop_matrix2d
pop_matrix3d
push_matrix2d
push_matrix3d
read_locator_event
rectangle
replace_matrix2d
replace_matrix3d
request_locator
sample_locator
set_locator
set_pick_depth
set_pick_window
shade_mode
shade_range
spline_curve2d
spline_curve3d
spline_surface
surface_model
text2d
text3d
text_alignment
text_line_path
text_line_space
text_orientation2d
text_orientation3d
transform_point
transform_points
u_knot_vector
v_knot_vector
vdc_extent
vdc_to_dc
vdc_to_wc
vertex_format
view_camera
view_matrix2d
view_matrix3d
view_port
view_volume
view_window
viewpoint
wc_to_vdc
zbuffer_switch

Integer only:

file_to_intbitmap
intarc
intbitmap_print
intbitmap_to_file
intblock_move
intblock_read
intblock_write
intcharacter_height
intcharacter_width
intcircle
intclip_rectangle
intconcat_matrix2d
intconcat_transform2d
intdraw2d
intecho_type2d
intecho_update2d
inthatch_spacing
intinquire_current_position2d
intinquire_pick_window
intinquire_text_extent2d
intline_repeat_length
intline_width
intmove2d
intpartial_arc
intpartial_circle
intpartial_polygon2d
intperimeter_repeat_length
intpolygon2d
intpolyline2d
line_endpoint
intpolycircle
intpolyrectangle
intpop_matrix2d
intpush_matrix2d
intrectangle
intreplace_matrix2d
intrequest_locator2d
intsample_locator2d
intset_pick_window
inttext2d
inttext_orientation2d
inttransform_point2d
intvdc_extent
intview_matrix2d
intview_port
intview_window

Device coordinate only:

dcbitmap_print
dcbitmap_to_file
dcblock_move
dcblock_read
dcblock_write
dccharacter_height
dccharacter_width
dccircle
dcdraw
dcecho_type
dcecho_update
dcmarker_size
dcmove
dcpartial_polygon
dcpolycircle
dcpolygon
dcpolyline
dcpolymarker
dcpolyrectangle
dcrectangle
dctext
file_to_dcbitmap

Common to Floating point and Integer:

character_expansion_factor
character_slant
clip_indicator
curve_resolution
flush_matrices
inquire_hit
intra_character_space
mapping_mode
pop_matrix
push_vdc_matrix
set_hit_mode
set_p1_p2
text_path
text_precision
vdc_justification

Common to all systems:

await_event
await_retrace
background_color
background_color_index
bank_switch
buffer_mode
clear_control
clear_view_surface
dbuffer_switch
define_color_table
define_raster_echo
designate_character_set
disable_events
display_enable
double_buffer
drawing_mode
enable_events
file_print
fill_color
fill_color_index
fill_dither
flush_buffer
gclose
gerr_message
gerr_print_control
gerr_procedure
gescape
gopen
hatch_orientation
hatch_type
initiate_request
inquire_display_mode
inquire_color_table
inquire_fb_configuration
inquire_gerror
inquire_id
inquire_input_capabilities
inquire_request_status
inquire_sizes
interior_style
line_color
line_color_index
line_type
make_picture_current
make_X11_gopen_string
marker_color
marker_color_index
marker_type
pattern_define
perimeter_color
perimeter_color_index
perimeter_type
read_choice_event
request_choice
rgb_to_index
sample_choice
set_signals
text_color
text_color_index
text_font_index
text_switching_mode
track
track_off
write_enable

HARDWARE DEPENDENCIES

There are two versions of the Starbase library file, libsb1.a, on the series 300 releases. (There is only one version, *libsb1.a*, on the series 800.) One series 300 version has been compiled for the 310 computer and will work for the 310, 318, 319, 320, 330, 340, 350, 360, or 370 and is named */usr/lib/libsb1.10.a*. The other series 300 version has been compiled to utilize 68020 instructions and to use the 68881 math co-processor. This version is named */usr/lib/libsb1.20.a* and will work on the 320, 330, 340, 350, 360, or 370 computers. The Starbase customize script, which is run whenever a system update is performed, links (ln(1)) the appropriate version into the */usr/lib/libsb1.a* file. If you are running in a clustered environment, */usr/lib/libsb1.a* is changed into a context-dependent file and the libraries *libsb1.10.a* and *libsb1.20.a* are linked to */usr/lib/libsb1.a+/HP-MC68010* and */usr/lib/libsb1.a+/HP-MC68881* , respectively. Depending on the context of the node, the proper library is automatically used during the compilation process.

To link a program that is to run on another system, it may be necessary to link with either libsb1.10.a or libsb1.20.a instead of libsb1.a in order to use the version best-suited for the target machine.

MKNOD SPECIAL FILES

HPGL, CADplt, and CADplt2 Device Drivers

The <Sc> parameter is the select code of the HP_IB or Serial Interface card. The <AD> parameter is the HP_IB address. The <LU> parameter is the logical unit of the HP_IB or Serial Interface card. The <Pt> parameter is the serial port address.

Series 300 Syntax

```
HPIB: mknod /dev/hpgl  c 21 0x<Sc><AD>00
Serial: mknod /dev/hpgls c  1 0x<Sc><Pt>04
```

Series 800 Syntax

```
HPIB: mknod /dev/hpgl  c 21 0x00<LU><AD>
Serial: mknod /dev/hpgls c  1 0x00<LU><Pt>
```

HP_HIL Device Driver

Series 300 Syntax

The <AD> parameter is the Address on the HIL serial loop.

```
mknod /dev/hilx c 24 0x0000<AD>
```

Series 800 Syntax

The <LU> parameter is the hardware logical unit number. The <AD> parameter is the device's HIL serial loop address.

```
mknod /dev/hilx c 24 0x00<LU><AD>
   or
mknod /dev/hil_<LU>.<AD> c 24 0x00<LU><AD>
```

HPTERM Device Driver

Use device file **/dev/tty** if you want the output to go to the terminal you are logged in on. See your System Administrator for details on using other terminals. Other terminals are usually designated as */dev/tty01*, */dev/tty02*, etc.

HP 300h/HP 300l Device Driver

Series 300 Syntax

```
mknod /dev/crt c 12 0x000000 ... (when alone)
mknod /dev/crt c 12 0x000100 ... (when also using hp98204b)
```

Series 800 Syntax

This driver is not supported on the series 800.

HP 9836A Device Driver

Series 300 Syntax

mknod /dev/crt c 12 0x000000

Series 800 Syntax

This driver is not supported on the series 800.

HP 98550 Device Driver

Series 300 Syntax

For internal devices:

mknod /dev/crt c 12 0x000000 *(for all planes)*
mknod /dev/ocrt c 12 0x000001 *(for overlay planes)*
mknod /dev/icrt c 12 0x000002 *(for image planes)*

For external device, <Sc> is the external select code of the HP 98548/49/50 devices.

mknod /dev/crt c 12 0x<Sc>0200 *(for all planes)*
mknod /dev/ocrt c 12 0x<Sc>0201 *(for overlay planes)*
mknod /dev/icrt c 12 0x<Sc>0202 *(for image planes)*

Series 800 Syntax

The <LU> parameter is the hardware logical unit number.

For internal devices:

mknod /dev/crt c 14 0x00<LU> *(for all planes)*
mknod /dev/ocrt c 14 0x00<LU>01 *(for overlay planes)*
mknod /dev/icrt c 14 0x00<LU>02 *(for image planes)*

HP 98556 Device Driver

Series 300 Syntax

The <Sc> parameter is the external select code of the HP 98556 device.

mknod /dev/crt c 12 0x<Sc>0200 *(for all planes)*
mknod /dev/ocrt c 12 0x<Sc>0201 *(for overlay planes)*
mknod /dev/icrt c 12 0x<Sc>0202 *(for image planes)*

Series 800 Syntax

The <LU> parameter is the logical unit number of the A1020A graphics subsystem.

mknod /dev/crt c 14 0x00<LU>00
mknod /dev/ocrt c 14 0x00<LU>01
mknod /dev/icrt c 14 0x00<LU>02

HP 98700 Device Driver

Series 300 Syntax

The <Sc> parameter is the external select code of the HP 98700 device.

mknod /dev/crt c 12 0x000000 ... (for console device)
mknod /dev/crt c 12 0x<Sc>0200 ... (for external select code)

Series 800 Syntax

This driver is not supported on the series 800.

HP 98710 Device Driver

Series 300 Syntax

The <Sc> parameter is the external select code of the HP 98710 device.

```
mknod /dev/crt c 12 0x000000 ... (for console device)
mknod /dev/crt c 12 0x<Sc>0200 ... (for external select code)
```

Series 800 Syntax

This driver is not supported on the series 800.

HP 98720 Device Driver

Series 300 Syntax

The <Sc> parameter is the external select code of the HP 98720 device.

```
mknod /dev/crt c 12 0x000000 ... (for console device)
mknod /dev/crt c 12 0x<Sc>0200 ... (for external select code)
```

Series 800 Syntax

The <LU> parameter is the hardware logical unit number.

```
mknod /dev/crt c 14 0x00<LU>00
```

HP 98720w Device Driver

Series 300 Syntax

The <Sc> parameter is the external select code of the HP 98720w device.

```
mknod /dev/ocrt c 12 0x000001 ... (for console device)
mknod /dev/ocrt c 12 0x<Sc>0201 ... (for external select code)
```

Series 800 Syntax

The <LU> parameter is the hardware logical unit number.

```
mknod /dev/crt c 14 0x00<LU>00
```

HP 98721 Device Driver

Series 300 Syntax

The <Sc> parameter is the external select code of the HP 98721 device.

```
mknod /dev/crt c 12 0x000000 ... (for console device)
mknod /dev/crt c 12 0x<Sc>0200 ... (for external select code)
```

Series 800 Syntax

The <LU> is the hardware logical unit number.

```
mknod /dev/crt c 14 0x00<LU>00
```

HP 98730 Device Driver

Series 300 Syntax

The <Sc> parameter is the external select code of the HP 98730 device.

```
mknod /dev/crt c 12 0x000000 ... (for console device)
mknod /dev/crt c 12 0x<Sc>0200 ... (for external select code)
```

Series 800 Syntax

The <LU> parameter is the hardware logical unit number.

```
mknod /dev/crt c 14 0x00<LU>00
```

HP 98731 Device Driver

Series 300 Syntax

The <Sc> parameter is the external select code of the HP 98731 device.

```
mknod /dev/crt c 12 0x000000 ... (for console device)
mknod /dev/crt c 12 0x<Sc>0200 ... (for external select code)
```

Series 800 Syntax

The <LU> is the hardware logical unit number.

```
mknod /dev/crt c 14 0x00<LU>00
```

HP Keyboard Device Driver

Locator Keyboard Device Driver

Use device file **/dev/tty** if you want the input to come from the terminal you are logged in on. See your System Administrator for details on using other terminals. Other terminals are usually designated as */dev/tty01*, */dev/tty02*, etc.

ENVIRONMENT VARIABLES

Series 300 only

SB_DISPLAY_ADDR

Frame buffer displays and the shared-memory communication mechanisms use a range of addresses on the series 300 machines. These addresses may conflict with extremely large code, heap, or stack space. This can be avoided by setting the environment variable SB_DISPLAY_ADDR to shift this range of addresses up or down.

If the Starbase Display Address environment variable (SB_DISPLAY_ADDR) is not set or is null, the default of 0xb00000 is used. This environment variable must be the same for all processes, so it is suggested that, if it is set, it be set in the Bourne file **/etc/profile** and the C Shell **/etc/csh.login**.

SEE ALSO

cc(1), fc(1), ld(1), pc(1), *Starbase Device Drivers Library Manual, Starbase Reference, Starbase Device Drivers Library Manual, Starbase Graphics Techniques, Starbase Programming with X11.*

NAME

surface_coefficients, bf_surface_coefficients – select the ambient, diffuse and specular coefficients for filled area primitives, and backfacing elements of filled area primitives

SYNOPSIS

C Syntax:

```
void surface_coefficients(fildes,ka,kd,ks);
int fildes;
float ka,kd,ks;

void bf_surface_coefficients(fildes,ka,kd,ks);
int fildes;
float ka,kd,ks;
```

FORTRAN77 Syntax:

```
subroutine surface_coefficients(fildes,ka,kd,ks)
integer*4 fildes
real ka,kd,ks

subroutine bf_surface_coefficients(fildes,ka,kd,ks)
integer*4 fildes
real ka,kd,ks
```

Pascal Syntax:

```
procedure surface_coefficients(fildes:integer;ka,kd,ks:real);

procedure bf_surface_coefficients(fildes:integer;ka,kd,ks:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

ka coefficient of ambient reflection

kd coefficient of diffuse reflection

ks coefficient of specular reflection

Discussion

The *surface_coefficients* subroutine specifies ambient, diffuse, and specular lighting constants for polygonal surfaces. *bf_surface_coefficients* specifies the same constants for backfacing shaded polygons. Its parameters are used to determine the intensity of backfacing shaded polygons. The use of surface coefficients on polygons and backfacing polygons is initiated by a call to *shade_mode*, turning shading on and to *bf_control*, turning on special handling for backfacing polygons.

The reflection coefficients defined here are multiplied by each of these terms and are used as a means to attenuate the contribution of various components. When shading polygons, the basic light source computations consist of the sum of three separate components:

The energy due to ambient illumination:

$$Ea = (ka)(Rs)(Ia)$$

The energy due to diffuse reflection from a point source, where n is device dependent (see discussion in *light_switch(3G)* on number of lights supported):

$$Ed = (kd)\sum_{j=1}^{n}(Rs)(Ip_j)(cos(i_j))$$

The energy due to specular reflection from a point source, where n is device dependent (see discussion in *light_switch(3G)* on number of lights supported):

$$Es = (ks)\sum_{j=1}^{n}(W)(Ip_j)((cos(s_j))^h)$$

Where:

Rs = Reflectance coefficient of the surface (RGB color set by *fill_color*)
Ia = Ambient illumination (RGB color set by *light_ambient*)
Ip = Point source illumination (RGB color set by *light_source*)
    ka = Coefficient of ambient reflection
    kd = Coefficient of diffuse reflection
    ks = Coefficient of specular reflection
i = Incident angle of light source
W = Specular reflection coefficient (*red, green, blue* color set by *surface_model*)
s = Angle between viewpoint ray and reflected ray.
h = Specular highlight power is degree of shininess (*highlight*)

DEFAULTS

ka = 1.0 kd = 1.0 ks = 1.0

SEE ALSO

bf_control(3G), fill_color(3G), interior_style(3G), perimeter_color(3G), surface_model(3G), *Starbase Graphics Techniques.*

NAME

surface_model, bf_surface_model – define surface light reflectance parameters for shaded polygon fill, and for backfacing elements of shaded polygons

SYNOPSIS

C Syntax:

```
void surface_model(fildes,specular,highlight,red,green,blue);
int fildes,specular,highlight;
float red,green,blue;

void bf_surface_model(fildes,specular,highlight,red,green,blue);
int fildes,specular,highlight;
float red,green,blue;
```

FORTRAN77 Syntax:

```
subroutine surface_model(fildes,specular,highlight,red,green,blue)
integer*4 fildes,specular,highlight
real red,green,blue

subroutine bf_surface_model(fildes,specular,highlight,red,green,blue)
integer*4 fildes,specular,highlight
real red,green,blue
```

Pascal Syntax:

```
procedure surface_model(fildes,specular,highlight:integer; red,green,blue:real);

procedure bf_surface_model(fildes,specular,highlight:integer; red,green,blue:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

specular If TRUE (1), specular reflections are calculated. If FALSE (0), they are not.

highlight Specular highlight power in the range of 1 to 16,383.

red, green, blue
Specular reflection coefficients in the range 0.0 to 1.0

Discussion

Surface_model specifies the specular light reflectance model of a polygonal surface, and the *bf_surface_model* subroutine specifies the specular light reflectance of backfacing polygonal surfaces. Parameters are used to determine the intensity of shaded polygons and backfacing shaded polygons. The use of the surface model on polygons and backfacing polygons is initiated by a call to *shade_mode*, turning shading on, and to *bf_control*, turning on special handling for backfacing polygons.

Specular reflection coefficients determine the color of the surface in the region of greatest specular reflection when reflecting a white light source. When modelling plastic surfaces the red, green and blue parameters will all be the same value as the specular reflection of most plastics is the color of the light source. This is generally not the case with metals.

When shading polygons, the basic light source computations consist of the sum of three separate components:

The energy due to ambient illumination:

$$Ea = (ka)(Rs)(Ia)$$

The energy due to diffuse reflection from a point source, where n is device dependent (see discussion in *light_switch(3G)* on number of lights supported):

$$Ed = (kd)\sum_{j=1}^{n}(Rs)(Ip_j)(cos(i_j))$$

The energy due to specular reflection from a point source, where n is device dependent (see discussion in *light_switch(3G)* on number of lights supported):

$$Es = (ks)\sum_{j=1}^{n}(W)(Ip_j)((cos(s_j))^h)$$

Where:

Rs = Reflectance coefficient of the surface (RGB color set by *fill_color*)
Ia = Ambient illumination (RGB color set by *light_ambient*)
Ip = Point source illumination (RGB color set by *light_source*)
ka = Coefficient of ambient reflection
kd = Coefficient of diffuse reflection
ks = Coefficient of specular reflection
i = Incident angle of light source
W = Specular reflection coefficient (*red, green, blue* color set by *surface_model*)
s = Angle between viewpoint ray and reflected ray.
h = Specular highlight power is degree of shininess (*highlight*)

If **specular** is FALSE (0), *Es* is not calculated.

Reflectance coefficients can also be specified directly for each polygon vertex if *vertex_format* specifies red, green, blue per vertex.

DEFAULTS

specular = FALSE (0): specular reflections not calculated.

SEE ALSO

bf_control(3G), fill_color(3G), interior_style(3G), light_ambient(3G), light_source(3G), light_switch(3G), perimeter_color(3G), shade_mode(3G), surface_coefficients(3G), vertex_format(3G), *Starbase Graphics Techniques.*

NAME

inttext2d, text2d, text3d, dctext – output a string of characters.

SYNOPSIS

C Syntax:

```
void inttext2d(fildes,x,y,string,xform);
int fildes,x,y,xform;
char *string;

void text2d(fildes,x,y,string,xform,more);
int fildes,xform,more;
float x,y;
char *string;

void text3d(fildes,x,y,z,string,xform,more);
int fildes,xform,more;
float x,y,z;
char *string;

void dctext(fildes,dcx,dcy,string);
int fildes,dcx,dcy;
char *string;
```

FORTRAN77 Syntax:

```
subroutine inttext2d(fildes,x,y,string,xform)
integer*4 fildes,x,y,xform
character*(*) string

subroutine text2d(fildes,x,y,string,xform,more)
integer*4 fildes,xform,more
real x,y
character*(*) string

subroutine text3d(fildes,x,y,z,string,xform,more)
integer*4 fildes,xform,more
real x,y,z
character*(*) string

subroutine dctext(fildes,dcx,dcy,string)
integer*4 fildes,dcx,dcy
character*(*) string
```

Pascal Syntax:

```
procedure inttext2d(fildes,x,y:integer;string:string255;xform:integer);

procedure text2d(fildes:integer;x,y:real;string:string255;xform,more:integer);

procedure text3d(fildes:integer;x,y,z:real;string:string255;xform,more:integer);

procedure dctext(fildes,dcx,dcy:integer;string:string255);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

x, y, z Starting coordinate for the string in current units.

dcx, dcy Starting coordinate for the string in device units.

string ASCII string of characters terminated with a null character.

xform Enumerated value specifying the type of transformation to perform on the font coordinates:

VDC_TEXT	=	vdc
WORLD_COORDINATE_TEXT	=	world coordinate
TOS_TEXT	=	top of matrix stack
ANNOTATION_TEXT	=	world coordinate and vdc

more Set to TRUE (1) if more characters are to be buffered with the current text and FALSE (0) if no buffering is needed.

Discussion

Text2d, *text3d*, and *inttext2d* primitives draw characters starting at the virtual device coordinate current position or at the specified (x,y) or (x,y,z) coordinate. In *text2d* and *text3d*, if **more** is non-zero, characters are buffered, awaiting more text before text alignment and drawing are performed. With *text2d* and *text3d*, if **more** is zero, alignment and drawing are done immediately. For *inttext2d* and *dctext*, drawing is performed immediately. **Xform** specifies the type of transformation to be applied to the font coordinates.

If **xform** equals VDC_TEXT (zero), the current font transformation (*character_height*, *intcharacter_height*, *intcharacter_width*, *character_width*, etc.) is pre-concatenated with the vdc to device coordinate transformation and used as the font-to-device coordinate transformation.

If **xform** equals WORLD_COORDINATE_TEXT (one), the current font transformation is pre-concatenated with the transformation matrix on top of the matrix stack and used as the font-to-device coordinate transformation.

If **xform** equals TOS_TEXT (two), the top of the matrix stack is used as the font-to-device coordinate transformation matrix.

If **xform** equals ANNOTATION_TEXT (three), the current transformation is set to a coordinate system with its origin at the x,y[,z] coordinate specified (in modelling units) and with scaling set to VDC's. The current font transformation is pre-concatenated with this transformation matrix and used as the font to device coordinate transformation. This provides readable (screen parallel) text, independent of the modelling transformation in effect. Three dimensional text orientation may override this screen parallelism if any of its vectors are not parallel to the screen.

The Device-Coordinate text procedure draws characters starting at the device coordinate specified and only with character path right. All control characters are ignored except for space (octal 40). Only fonts 1 and 2 work correctly with *dctext*; the other fonts may or may not draw the entire character.

Only the Device-Coordinate character attributes (*dccharacter_height* and *dccharacter_width*) are used in drawing Device-Coordinate text.

The *inttext2d* procedure draws characters starting at the coordinates specified with the default text alignment for the specified path. All control characters are ignored except space (octal 40). Only fonts 1 and 2 work correctly with *inttext2d*; the other fonts may not draw the entire character.

The character attributes *intcharacter_height*, *intcharacter_width*, *inttext_orientation2d*, and *intinquire_text_extent2d* affect only *inttext2d*.

Procedures *text_alignment*, *text_line_path*, and *text_line_space* do not affect *inttext2d*.

Integer operations are only available when using the INT_XFORM *gopen* **mode**. When in INT_XFORM mode, floating point operations are not available for that **fildes**. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase(3G)* manual page.

SEE ALSO

append_text(3G), character_height(3G), text_alignment(3G), text_color(3G), text_font_index(3G), text_line_path(3G), text_line_space(3G), text_orientation(3G), text_path(3G), text_precision(3G), text_switching_mode(3G), *Starbase Graphics Techniques*.

NAME

text_alignment – set text line alignment relative to starting point of each line

SYNOPSIS

C Syntax:

```
void text_alignment(fildes,h_select,v_select,horizontal,
                    vertical);
int fildes,h_select,v_select;
float horizontal,vertical;
```

FORTRAN77 Syntax:

```
subroutine text_alignment(fildes,h_select,
                    v_select,horizontal,vertical)
integer*4 fildes,h_select,v_select
real horizontal,vertical
```

Pascal Syntax:

```
procedure text_alignment(fildes,h_select,v_select:integer;
horizontal,vertical:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

h_select A value in:

TA_LEFT
TA_CENTER
TA_RIGHT
TA_CONTINUOUS_HORIZONTAL
TA_NORMAL_HORIZONTAL

v_select A value in:

TA_TOP
TA_CAP
TA_HALF
TA_BASE
TA_BOTTOM
TA_CONTINUOUS_VERTICAL
TA_NORMAL_VERTICAL

horizontal Fraction of the face of the text extent box which appears on the negative side of the up vector. This value is only used with TA_CONTINUOUS_HORIZONTAL a be any real value.

vertical Fraction of the face of the text extent box which appears on the positive side of the base vector. This value is only used with TA_CONTINUOUS_VERTICAL and be any real value.

Discussion

Text_alignment specifies the position of the text rectangle relative to the point specified in the *text2d* or *text3d* function. If the **h_select** and/or the **v_select** parameter(s) is continuous, the text rectangle position is specified by the horizontal and/or vertical parameter(s). The normal values depend on the character path at the time the text primitive is called:

path	normal horizontal	normal vertical
right	left	baseline
left	right	baseline
up	center	baseline
down	center	top

TA_CENTER is equivalent to TA_CONTINUOUS_HORIZONTAL and **h_select** = 0.5.

TA_HALF is equivalent to TA_CONTINUOUS_VERTICAL and **v_select** = 0.5.

DEFAULTS

TA_NORMAL_HORIZONTAL and TA_NORMAL_VERTICAL.

SEE ALSO

text(3G), *Starbase Graphics Techniques.*

NAME

text_color – select color for subsequent text operations

SYNOPSIS

C Syntax:

```
void text_color_index(fildes,index);
int fildes,index;

void text_color(fildes,red,green,blue);
int fildes;
float red,green,blue;
```

FORTRAN77 Syntax:

```
subroutine text_color_index(fildes,index)
integer*4 fildes,index

subroutine text_color(fildes,red,green,blue)
integer*4 fildes
real red,green,blue
```

Pascal Syntax:

```
procedure text_color_index(fildes,index:integer);

procedure text_color(fildes:integer;red,green,blue:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

index is an integer index into the device color table. The text color is the red, green, and blue values specified at this table location. The size of the color table may be obtained using the *inquire_sizes* procedure.

red, green, blue Color values (in the range of 0.0 to 1.0) to be used for subsequent text primitives. Zero (0.0) indicates **no** color and one (1.0) indicates **full** color. Monochrome devices use the equation

$$I = 0.30(\text{red}) + 0.59(\text{green}) + 0.11(\text{blue})$$

to determine intensity.

Discussion

To acquire the current color table definition, use the *inquire_color_table* command. To change the current color table definition, use the *define_color_table* procedure.

Text primitives are altered by this procedure.

When text color is set using red, green, and blue parameters, determination of actual output depends on the state of the mode parameter set with *shade_mode*:

CMAP_NORMAL — This is the default mode if *shade_mode* has not been called. The color table is searched for an index which points to the closest color in RGB space to the one specified. Specification by index is more efficient than color map searches.

CMAP_MONOTONIC — Color values are converted to intensity using the equation:

$$I = 0.30(\text{red}) + 0.59(\text{green}) + 0.11(\text{blue}).$$

This intensity is mapped to an index, using the minimum and maximum defined by *shade_range*.

CMAP_FULL — Color values are mapped directly to an index with the assumption that the color map is set up to a predefined full color state.

If the color map is changed after this procedure is called, the text color used may not be the original color desired.

If the index is out of range, a warning is generated and a mod function is performed.

DEFAULTS

Default text color is the color defined in the current color map at index 1.

SEE ALSO

define_color_table(3G), inquire_color_table(3G), shade_mode(3G), text(3G).

NAME

text_font_index – select character font for subsequent text primitives

SYNOPSIS

C Syntax:

```
void text_font_index(fildes,index);
int fildes,index;
```

FORTRAN77 Syntax:

```
subroutine text_font_index(fildes,index)
integer*4 fildes,index
```

Pascal Syntax:

```
procedure text_font_index(fildes,index:integer);
```

DESCRIPTION

INPUT PARAMETER

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

index Font file designation to be appended to the character-set path name.

Discussion

Text_font_index selects a font for the currently designated character set. The font is at the concatenation of the character-set pathname and the text-font index name, where the character-set pathname is the active character-set directory.

Predefined character sets and fonts are located at
/usr/lib/starbase/stroke/'chset name'/'font index'

Use *ls (1)* on the appropriate directory to see what fonts are available. For example, to see what fonts are available for the character set **usascii** do an *ls* of the directory: */usr/lib/starbase/stroke/usascii*. Font 1 is a fixed-width font, while font 2 is a variable-width font.

DEFAULTS

Font **index** = 1 (as set in */usr/lib/starbase/defaults*).

SEE ALSO

designate_character_set(3G), *Starbase Graphics Techniques.*

NAME

text_line_path – define relative position between successive lines of text.

SYNOPSIS

C Syntax:

```
void text_line_path(fildes,path);
int fildes,path;
```

FORTRAN77 Syntax:

```
subroutine text_line_path(fildes,path)
integer*4 fildes,path
```

Pascal Syntax:

```
procedure text_line_path(fildes,path:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

path One of the following:

PATH_RIGHT
PATH_LEFT
PATH_UP
PATH_DOWN

Discussion

Text_line_path selects the direction between subsequently drawn text strings (the direction of the line-feed character):

PATH_RIGHT:	along the base vector.
PATH_LEFT:	180 degrees from the base vector.
PATH_UP:	along the up vector.
PATH_DOWN:	180 degrees from the up vector.

DEFAULTS

PATH_DOWN

SEE ALSO

Starbase Graphics Techniques.

NAME

text_line_space – set spacing between lines for subsequent text procedures

SYNOPSIS

C Syntax:

```
void text_line_space(fildes,space);
int fildes;
float space;
```

FORTRAN77 Syntax:

```
subroutine text_line_space(fildes,space)
integer*4 fildes
real space
```

Pascal Syntax:

```
procedure text_line_space(fildes:integer;space:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

space Fraction of the character's height to be added between lines.

Discussion

Text_line_space specifies the space between lines as a fraction of the character height. The space is added along the line path.

DEFAULTS

space = 0.0

SEE ALSO

Starbase Graphics Techniques.

NAME

inttext_orientation2d, text_orientation2d, text_orientation3d – specify text orientation.

SYNOPSIS

C Syntax:

```
void inttext_orientation2d(fildes,up_x,up_y,base_x,base_y);
int fildes,up_x,up_y,base_x,base_y;

void text_orientation2d(fildes,up_x,up_y,base_x,base_y);
int fildes;
float up_x,up_y,base_x,base_y;

void text_orientation3d(fildes,up_x,up_y,up_z,base_x,base_y,base_z);
int fildes;
float up_x,up_y,up_z,base_x,base_y,base_z;
```

FORTRAN77 Syntax:

```
subroutine inttext_orientation2d(fildes,up_x,up_y,base_x,base_y)
integer*4 fildes,up_x,up_y,base_x,base_y

subroutine text_orientation2d(fildes,up_x,up_y,base_x,base_y)
integer*4 fildes
real up_x,up_y,base_x,base_y

subroutine text_orientation3d(fildes,up_x,up_y,
                up_z,base_x,base_y,base_z)
integer*4 fildes
real up_x,up_y,up_z,base_x,base_y,base_z
```

Pascal Syntax:

```
procedure inttext_orientation2d(fildes,up_x,up_y,base_x,base_y:integer);

procedure text_orientation2d(fildes:integer;up_x,up_y,base_x,base_y:real);

procedure text_orientation3d(fildes:integer;up_x,up_y,up_z,base_x,base_y,
                base_z:real);
```

DESCRIPTION

Input Parameters

fildes	Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.
up_x,up_y,up_z	Coordinates of the up vector relative to (0.0,0.0,0.0). Vector is specified in vdc coordinates.
base_x,base_y,base_z	Coordinates of the base vector relative to (0.0,0.0,0.0). Vector is specified in vdc coordinates.

Discussion

Text_orientation defines two vectors which, in turn, determine the orientation of subsequently drawn text characters. The up vector has the effect of slanting characters away from the vertical, and also defines the direction of line feed characters. The base vector defines the positioning of subsequent characters in a string when the text path is either left or right.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

DEFAULTS

Floating point:
up_x,up_y,up_z = (0.0,1.0,0.0)

base_x,base_y,base_z = (1.0,0.0,0.0)

Integer:
up_x,up_y, = (0,1)
base_x,base_y = (1,0)

SEE ALSO
Starbase Graphics Techniques.

NAME

text_path – selects the direction of subsequently drawn text characters.

SYNOPSIS

C Syntax:

```
void text_path(fildes,path);
int fildes,path;
```

FORTRAN77 Syntax:

```
subroutine text_path(fildes,path)
           integer*4 fildes,path
```

Pascal Syntax:

```
procedure text_path(fildes,path:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

path One of the values:

PATH_RIGHT
PATH_LEFT
PATH_UP
PATH_DOWN

Discussion

Text_path selects the writing direction of subsequently drawn text characters:

PATH_RIGHT:	Along the base vector
PATH_LEFT:	180 degrees from the base vector
PATH_UP:	Along the up vector
PATH_DOWN:	180 degrees from the up vector

DEFAULTS

PATH_RIGHT

SEE ALSO

Starbase Graphics Techniques.

NAME

text_precision – select how text will be drawn.

SYNOPSIS

C Syntax:

```
void text_precision(fildes,precision);
int fildes, precision;
```

FORTRAN77 Syntax:

```
subroutine text_precision(fildes,precision)
integer*4 fildes,precision
```

Pascal Syntax:

```
procedure text_precision(fildes,precision:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

precision One of the values

STRING_TEXT
CHARACTER_TEXT
STROKE_TEXT

Discussion

Text_precision selects the method by which subsequent text strings are drawn. If STRING_TEXT is selected, the device's text capabilities are used. Hardware text generation, in general, provides only limited approximations to the actual text-size attributes which may have been set. Text attributes for hardware text may need to be set directly on the device by sending escape codes using a GESCAPE call. Consult the device's reference manual for a list of these escape codes.

STROKE_TEXT causes characters to be drawn at a higher precision than hardware text, but at a slower rate on some devices.

Currently, STROKE_TEXT precision is used for CHARACTER_TEXT precision.

DEFAULTS

precision = STROKE_TEXT

SEE ALSO

Starbase Graphics Techniques.

NAME

text_switching_mode – select text character set designation and invocation mode

SYNOPSIS

C Syntax:

```
void text_switching_mode(fildes,mode);
int fildes,mode;
```

FORTRAN77 Syntax:

```
subroutine text_switching_mode(fildes,mode)
integer*4 fildes,mode
```

Pascal Syntax:

```
procedure text_switching_mode(fildes,mode:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

mode Specifies text set designation mode.

Discussion

There are several specifiable text switching modes:

ISO_7BIT mode - (**mode** =0) similar to ISO 8-bit mode but only the lower seven bits of each character are used.

ISO_8BIT mode - (**mode** = 1) implements ISO 2022.2(1982), permitting designation and invocation of all four G-sets individually.

HP_8BIT mode - (**mode** = 2) implements the HP 8-bit standard. Designating character set G0 automatically designates its associated set, G2. Designating character set G1 automatically designates its associated set, G3. The associated set is the set used when the eighth bit of a character is set. An associated font can use the HP-15 format for character sets that need 2-bytes/character. An HP-15 font should not be used with other switching modes.

Designating **usascii** is the same as using the HP-UX character set ROMAN8, and **jisascii** is the same as the HP-UX set KANA8.

The following is a list of each font, and its associated set:

Font Name	Associated Set
usascii	HPROMAN
hproman	NULL
jisascii	KATAKANA
katakana	NULL
jpn	KANJI
kanji	KATAKANA

Fonts for the various character sets are located in */usr/lib/starbase/stroke/font*.

The escape codes needed to work with the character sets and fonts are listed in the table below.

Shift Function		ISO_7BIT	ISO_8BIT	HP_8BIT
Shift out	SO	0/14	-	0/14
Shift in	SI	0/15	-	0/15
Locking shift zero	LS0	-	00/15	-
Locking shift one	LS1	-	00/14	-
Locking shift one right	LS1R	-	esc 7/14	-
Locking shift two	LS2	esc 6/14	esc 6/14	-
Locking shift two right	LS2R	-	esc 7/13	-
Locking shift three	LS3	esc 6/15	esc 6/15	-
Locking shift three right	LS3R	-	esc 7/12	-
Single shift two	SS2	esc 4/14	08/14	-
Single shift three	SS3	esc 4/15	08/15	-

If a single-byte representation of SS2 is required in 7 bits, it should be coded as 1/9.

DEFAULTS

ISO_8BIT mode as set in */usr/lib/starbase/defaults*.

SEE ALSO

designate_character_set(3G)

NAME

track – asynchronously echo an input device's locator position on an output device.

SYNOPSIS

C Syntax:

```
void track(indev,outdev,locator_num);
int indev,outdev,locator_num;
```

FORTRAN77 Syntax:

```
subroutine track(indev,outdev,locator_num)
integer*4 indev,outdev,locator_num
```

Pascal Syntax:

```
procedure track(indev,outdev,locator_num:integer);
```

DESCRIPTION

Input Parameters

indev File descriptor for the input device whose locator subdevice is to be tracked.

outdev File descriptor for the output device whose echo is to be used to track the input device.

locator_num Number of the locator subdevice on the specified input device whose position is to be tracked (the number of locators on a device may be obtained using the *inquire_sizes* procedure).

Discussion

Track sets up asynchronous tracking of an input device's locator position onto an output device's view surface. The locator position in virtual device coordinates is mapped onto the virtual device coordinates of the output device. If the vdc extents of the two devices are not similar, the tracking movement will be either exaggerated or reduced.

When done tracking, *track_off(3G)* should be called. This turns off the tracking daemon and cleans up the ipc so the tracking process can communicate with the running Starbase program.

SEE ALSO

track_off(3G)

NAME

track_off – stop asynchronous tracking

SYNOPSIS

C Syntax:

```
void track_off(fildes);
int fildes;
```

FORTRAN77 Syntax:

```
subroutine track_off(fildes)
integer*4 fildes
```

Pascal Syntax:

```
procedure track_off(fildes:integer);
```

DESCRIPTION

INPUT PARAMETER

fildes Integer file descriptor returned by *gopen* when the I/O path to a graphics device is opened.

Discussion

Track_off stops the asynchronous tracking started by *track*. **Fildes** is associated with the input device being tracked.

SEE ALSO

track(3G)

NAME

inttransform_point2d, transform_point, transform_points – transform a point or points from one coordinate system to another

SYNOPSIS

C Syntax:

```
void inttransform_point2d(fildes,mode,inx,iny,outx,outy);
int fildes,mode,inx,iny,*outx,*outy;

void transform_point(fildes,mode,inx,iny,inz,outx,outy,outz);
int fildes,mode;
float inx,iny,inz,*outx,*outy,*outz;

void transform_points(fildes,wc_pts,dc_pts,numpts,homogeneous);
int fildes,numpts,homogeneous;
float wc_pts[],dc_pts[];
```

FORTRAN77 Syntax:

```
subroutine inttransform_point2d(fildes,mode,inx,iny,outx,outy)
integer*4 fildes,mode,inx,iny,outx,outy

subroutine transform_point(fildes,mode,inx,iny,inz,outx,outy,outz)
integer*4 fildes,mode
real inx,iny,inz,outx,outy,outz

subroutine transform_points(fildes,wc_pts, dc_pts,num_pts,homogeneous)
integer*4 fildes,numpts,homogeneous
real wc_pts(numpts*3),dc_pts(numpts*4)
or real wc_pts(numpts*4),dc_pts(numpts*4)
```

Pascal Syntax:

```
procedure inttransform_point2d(fildes,mode,inx,iny:integer;
        var outx,outy:integer);

procedure transform_point(fildes,mode:integer;inx,iny,inz:real;
        var outx,outy,outz:real);

procedure transform_points(fildes:integer;
        var wc_pts,dc_pts:array[lo..hi:integer]of real;
        numpts,homogeneous:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

mode Defines the transformation to be made (*transform_point*):

- MC_TO_WORLD transform from modelling coordinates to world coordinates.
- MC_TO_VDC transform from modelling coordinates to virtual device coordinates.
- WORLD_TO_VDC transform from world coordinates to virtual device coordinates.
- WORLD_TO_MC transform from world coordinates to modelling coordinates.
- VDC_TO_WORLD transform from virtual device coordinates to world coordinates.
- VDC_TO_MC transform from virtual device coordinates to modelling coordinates.

INTVDC_TO_DC transform from virtual device coordinates to device coordinates.
DC_TO_INTVDC transform from device coordinates to virtual device coordinates.

inx,iny,inz Point to be transformed (*transform_point*).

wc_pts Array of world-coordinate points to be transformed (*transform_points*).

num_pts Number of points to be transformed (*transform_points*).

homogeneous TRUE if the **wc_pts** array contains homogeneous coordinates (*transform_points*).

Output Parameters

outx,outy,outz Transformed point (*transform_point*).

dc_pts Array of transformed points in device coordinates (*transform_points*).

Discussion:

Transform_point and *inttransform_point2d* transform a single point from one coordinate system to another, using the current set of transformation matrices. *Transform_point* and *inttransform_point2d* left-multiply a coordinate vector made up of **inx,iny,inz** or **inx,iny** with the appropriate transformation matrix, as specified by **mode**, and returns the resulting transformed point in the coordinate vector made up of **outx,outy,outz** or **outx,outy**.

The coordinate systems used by Starbase can be conceptually defined as follows:

1. User points are assumed to be defined in modelling coordinates. These points are transformed into world coordinates using a matrix called the **modelling transformation matrix** if any matrices have been pushed on the stack. World coordinates are used to perform any rendering calculations needed. Use matrix stack manipulation functions to place modelling transformation matrices on the matrix stack.
2. World coordinates are transformed to device coordinates by the **viewing transformation matrix**, which is usually the concatenation of user-defined viewing transformations and the vdc-to-device transformation matrix. When a device is opened, the **viewing transformation matrix** is simply the vdc-to-device coordinate transformation matrix. *Intview_window, intview_port, intview_matrix2d, view_window, view_port, view_volume, view_camera,* and *view_matrix* can be used to define further viewing transformations.

Use matrix stack manipulation functions to place modelling transformation matrices on the matrix stack. Use *view_matrix* to set viewing transformations.

- If a graphics device has been opened in MODEL_XFORM mode, the modelling transformation cannot be combined with the viewing transformation because special rendering calculations such as shading may be needed after the modelling-to-world-coordinate transformation. Therefore, matrices pushed on the stack are left undisturbed and all transformations from modelling coordinates to device coordinates are processed in two steps: modelling to world coordinates, followed by world to device coordinates.
- If a graphics device is not in MODEL_XFORM mode, the modelling and viewing transformations can be combined. Thus, the current viewing transformation matrix is post-concatenated to modelling matrices placed on the matrix stack, and subsequent output primitives are transformed, using only the top matrix on the matrix stack.

Transform_point and *inttransform_point2d* calculates matrix inverses only if a given transformation matrix has been changed since its inverse was last calculated. If in MODEL_XFORM mode, inverses are needed to perform the following transformations: WORLD_TO_MC, VDC_TO_WORLD, and VDC_TO_MC. If not in MODEL_XFORM mode, inverses are needed for the cases listed above, as well as MC_TO_WORLD.

Actually, the current viewing transformation matrix is post-concatenated to modelling matrices placed on the matrix stack, and subsequent output primitives are transformed, using only the top matrix on the matrix stack.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

Discussion: *transform_points*

Transform_points transforms an array of points, **wc_pts**, using the top matrix on the matrix stack to calculate transformed coordinate point values. The results are placed in array **dc_pts**. If the homogeneous (perspective) coordinate is zero, the data is three dimensional with 3 coordinates per point. Since Starbase uses (4 x 4) transformation matrices internally, a homogeneous coordinate value of 1 is assigned to points when the user supplies only 3 coordinates per point. If the homogeneous (perspective) coordinate is non-zero, there are 4 coordinates per point with the fourth being the homogeneous coordinate. Returned points, however, always have 4 coordinates per point. No perspective divide is performed. This function makes use of the floating-point hardware in the device, if present, and can be used for generalized (n x 4) * (4 x 4) transformations. If the device does not have floating point hardware and the device has been gopened in the 2D mode, the homogeneous coordinate is ignored.

Note that if the graphics device has been opened in MODEL_XFORM mode, the top matrix on the matrix stack is the modelling-to-world coordinate transformation. If modelling matrices have been pushed on the stack, the points will be transformed through the current modelling matrix only; not the viewing matrix. If the graphics device is not in MODEL_XFORM mode then the top matrix on the matrix stack is the concatenation of the modelling transformation with the viewing transformation and vdc-to-device matrix. Thus, points are transformed from modelling coordinates to device coordinates.

NOTE

When using *inttransform_point2d* to transform points between coordinate systems with large resolution differences, some precision may be lost during intermediate transformations. For example, the transformation from world coordinates to virtual device coordinates uses intermediate values in the device coordinate system. The precision of the virtual device coordinates result is therefore dependent on the coordinate system with the least resolution.

SEE ALSO

concat_matrix(3G), concat_transformation(3G), flush_matrices(3G), gopen(3G), pop_matrix(3G), push_matrix(3G), replace_matrix(3G), view_matrix(3G), *Starbase Graphics Techniques*.

NAME

triangular_strip – defines a series of triangular regions to be filled and/or edged.

SYNOPSIS

C Syntax:

```
void triangular_strip(fildes,clist,numverts,gnormals);
int fildes,numverts;
float clist[],gnormals[];
```

FORTRAN77 Syntax:

```
subroutine triangular_strip(fildes,clist,numverts,gnormals)
integer*4 fildes,numverts
real clist(numverts*3),gnormals((numverts-2)*3)
```

Pascal Syntax:

```
procedure triangular_strip(fildes:integer;var clist:array[lo..hi:integer] of real;
        numverts:integer;var gnormals:array[lo..hi:integer] of real);
```

DESCRIPTION

Input Parameters

fildes	Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.
clist	array of three dimensional real endpoint data
numverts	number of strip vertices in the clist array.
gnormals	array of real geometric normals.

Discussion

A boundary of a series of triangular regions is defined by connecting each vertex to its two successors in the coordinate list. The first triangle is formed from vertices 1,2,3 and the second triangle from vertices 2,3,4. The last triangle if formed from vertices numverts-2,numverts-1,numverts.

If *vertex_format* is specified to include a normal per polygon, then this normal is extracted from the *gnormals* array. The first three entries of this array are x,y,z coordinates of the normal for the first triangle. The second three entries are the normal for the second triangle and so on. If there are no normals per polygon then the gnormals value may be null.

Each triangle is filled and/or outlined according to the current interior style. *Triangular_strip* uses the current fill color and perimeter attributes. As with all output primitives it is affected by the current drawing mode and write enable.

SEE ALSO

drawing_mode(3G), fill_color(3G), interior_style(3G), perimeter_color(3G), perimeter_repeat_length(3G), perimeter_type(3G), polygon(3G), vertex_format(3G), write_enable(3G).

NAME

trimming_curve – define a spline-trimming curve

SYNOPSIS

C Syntax:

```
void define_trimming_curve (fildes,plist,numpts,order,rational);
int fildes, numpts, order, rational;
float plist[];
```

FORTRAN77 Syntax:

```
subroutine define_trimming_curve (fildes,plist,numpts,order,rational)
integer*4 fildes, numpts, order, rational
real plist (numpts*(2+rational))
```

Pascal Syntax:

```
procedure define_trimming_curve (fildes:integer;
var plist array [lo..hi:integer] of real;
numpts, order, rational:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

numpts Number of points in the plist array.

plist Array of spline control points in *u v* space.

order Trimming curve order:
LINEAR = 2, QUADRATIC = 3, CUBIC = 4, QUARTIC = 5, QUINTIC = 6.

rational Specifies whether the trimming curve is RATIONAL (1) or NONRATIONAL (0).

Discussion

define_trimming_curve specifies a spline curve of the specified order, using the points in the plist array as control points and the current *u* knot vector. The control points are specified in *u* and *v* coordinate space. All defined trimming curves are used for trimming the next surface drawn with the *spline_surface* procedure. These trimming curves are then deleted.

If trimming curves have been defined, a determination of whether the (umin,vmin) point of the *u v* mesh is visible or not is made by examining the first and last control points of each curve. If one of these control points is at (umin,vmin), the point is assumed to be visible. Otherwise, it is treated as invisible. When following a trimming curve from point to point, the visible portion of the surface is assumed to be on the left.

Combinations of trimming curves must form closed loops. One trimming curve joins the next if its last control point is equal to the first control point of the next trimming curve and if a u_knot vector has been defined, or bezier_knots are being used. If trimming curves intersect, unpredictable results occur. A surface-bounding trimming curve must be provided if the edge of the surface is visible and there are trimming curves interior to the surface.

DEFAULTS

No trimming curves are defined.

SEE ALSO

knot_vectors(3G), spline(3G), *Starbase Graphics Techniques*.

NAME

intvdc_extent, vdc_extent – define logical region of interest (window) for subsequent output primitives

SYNOPSIS

C Syntax:

```
void intvdc_extent(fildes,xmin,ymin,xmax,ymax);
int fildes,xmin,ymin,xmax,ymax;

void vdc_extent(fildes,xmin,ymin,zmin,xmax,ymax,zmax);
int fildes;
float xmin,ymin,zmin,xmax,ymax,zmax;
```

FORTRAN77 Syntax:

```
subroutine intvdc_extent(fildes,xmin,ymin,xmax,ymax)
integer*4 fildes,xmin,ymin,xmax,ymax

subroutine vdc_extent(fildes,xmin,ymin,zmin,xmax,ymax,zmax)
integer*4 fildes
real xmin,ymin,zmin,xmax,ymax,zmax
```

Pascal Syntax:

```
procedure intvdc_extent(fildes,xmin,ymin,xmax,ymax:integer);

procedure vdc_extent(fildes:integer;xmin,ymin,zmin,xmax,ymax,zmax:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when the I/O path to a graphic device is opened.

xmin,ymin,zmin,xmax,ymax,zmax
Define, respectively, the lower-left front and upper-right back corners of the virtual device coordinate extent.

Discussion

Vdc_extent defines a logical Cartesian coordinate system for use in transforming virtual device coordinates to physical device coordinates.

This coordinate system establishes the sense and orientation of the virtual device coordinate extent. Sense and orientation are defined by the directions of the positive x and y axes and whether the positive y-axis is 90 degrees clockwise or 90 degrees counter-clockwise from the positive x-axis. The coordinate system also establishes the direction of positive and negative angles. Positive angles are measured in the direction from positive x-axis to positive y-axis.

The meanings of **up**, **down**, **right**, **left**, **front**, and **back** are also defined (left-handed coordinate system):

Up	Positive y-axis
Down	Negative y-axis
Right	Positive x-axis
Left	Negative x-axis
Back	Positive z-axis
Front	Negative z-axis

Specifying values outside the vdc extent area is permitted, but it is intended that the visible portion of the image be contained within the extent area. For example, when portraying part of the surface of a sphere, the center of the sphere could be outside the extent area. The extent area is intended to provide a frame for the region of interest in a picture.

The actual clipping bounds must always be a subset of the vdc system. This means that Starbase may truncate the clipping bounds if necessary.

Based on the current settings of other coordinate system parameters, the vdc extent uses as much of the physical region of interest as possible without distorting the image or eliminating any part of the image within the range of the extent. See *set_p1_p2*, *mapping_mode*, and *vdc_justification* for ways to modify this mapping.

When outputting to a graphics device, if the current extent values match the device units, no transformation occurs, making the output process more efficient.

If **xmin** equals **xmax** or **ymin** equals **ymax** an error is generated and the vdc extent is not changed.

This procedure updates the current vdc-to-device units transformation matrix. If no matrices have been placed on the matrix stack and no viewing transformations have been defined, the new vdc matrix becomes the current viewing transformation. No other transformation matrices are affected.

Changes to vdc extent may not alter previously set values such as text size or tracking relationships. Therefore, vdc extent should be set before setting related values. *Vdc_extent* and *intvdc_extent* resets *view_port* or *intview_port* to the entire vdc extent.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

Beware:

The *vdc_extent* function (beginning at HP-UX Release 5.18 and subsequently) changes the **clip_rectangle** and **clip_depth** values to be the same as the bounds of *vdc_extent*. In other words, **clip_rectangle** is set to be **xmin** to **xmax** in the x direction and **ymin** to **ymax** in the y direction and **clip_depth** is set to be **zmin** to **zmax**. This is a change from previous behavior where **clip_rectangle** and **clip_depth** were not changed by *vdc_extent*.

DEFAULTS

Floating point:
(xmin,ymin,zmin,xmax,ymax,zmax) = (0.0,0.0,0.0,1.0,1.0,1.0)
Integer:
(xmin,ymin,xmax,ymax) = (0,0,32767,32767)

SEE ALSO

clip_rectangle(3G), mapping_mode(3G), set_p1_p2(3G), vdc_justification(3G).

NAME

vdc_justification – control exact placement of VDC extent within physical region specified by *set_p1_p2* when *mapping_mode* is isotropic. This used to be called *viewport_justification*.

SYNOPSIS

C Syntax:

```
void vdc_justification(fildes,left,bottom);
int fildes;
float left,bottom;
```

FORTRAN77 Syntax:

```
subroutine vdc_justification(fildes,left,bottom)
integer*4 fildes
real left,bottom
```

Pascal Syntax:

```
procedure vdc_justification(fildes:integer;
left,bottom:real);
```

DESCRIPTION

Input Parameters

fildes — Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

left and **bottom** — Defines the fraction of "white space" on the left and bottom sides of the viewport.

Discussion

When *mapping_mode* is isotropic, the aspect ratio of the physical region of interest does **not** match the aspect ratio of the vdc extent. In other words, there will be extra "white space" inside the physical region of interest.

If the physical region of interest is wider than the vdc extent when the heights match, **left** specifies the fraction of the total white space that is inserted between P1 and the left side of the image of the vdc extent.

If the physical region of interest is taller than the vdc extent when the widths match, **bottom** specifies the fraction of the white space to be inserted between P1 and the bottom of the image of the vdc extent.

Left and **bottom** are decimal fractions between zero (0.0) and one (1.0), inclusive. Out-of-range parameter(s) generate a parameter error and the command is ignored.

This procedure updates the current vdc-to-device units transformation matrix. If no matrices have been placed on the matrix stack and no viewing transformations have been defined, the new vdc matrix becomes the current viewing transformation. Otherwise, no other transformation matrices are affected.

Note: This function was called viewport_justification in previous releases. For compatibility, there is still a viewport_justification function that just calls vdc_justification.

DEFAULTS

left = 0.5, **bottom** = 0.5

SEE ALSO

mapping_mode(3G), set_p1_p2(3G), vdc_extent(3G).

NAME

vdc_to_dc – transform a virtual device coordinate point to a device coordinate point using the current vdc-to-device coordinate transformation.

SYNOPSIS

C Syntax:

```
void vdc_to_dc(fildes,vdcx,vdcy,vdcz,dcx,dcy,dcz)
int fildes;
float vdcx,vdcy,vdcz;
int *dcx,*dcy,*dcz;
```

FORTRAN77 Syntax:

```
subroutine vdc_to_dc(fildes,vdcx,vdcy,vdcz,dcx,dcy,dcz)
integer*4 fildes
real vdcx,vdcy,vdcz
integer*4 dcx,dcy,dcz
```

Pascal Syntax:

```
procedure vdc_to_dc(fildes:integer;vdcx,vdcy,vdcz:real;
var dcx,dcy,dcz:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

vdcx,vdcy,vdcz
Defines a point in virtual device coordinates to be transformed into device coordinates.

Output Parameters

dcx,dcy,dcz Transformed point in device coordinates.

Discussion

Vdc_to_dc is used to transform a point in virtual device coordinate values to device coordinates.

SEE ALSO

dc_to_vdc(3G), transform_point(3G).

NAME

vdc_to_wc – transform virtual device coordinate point to world coordinate point using inverse of current transformation matrix

SYNOPSIS

C Syntax:

```
void vdc_to_wc(fildes,vdcx,vdcy,vdcz,wcx,wcy,wcz);
int fildes;
float vdcx,vdcy,vdcz,*wcx,*wcy,*wcz;
```

FORTRAN77 Syntax:

```
subroutine vdc_to_wc(fildes,vdcx,vdcy,vdcz,wcx,wcy,wcz)
integer*4 fildes
real vdcx,vdcy,vdcz,wcx,wcy,wcz
```

Pascal Syntax:

```
procedure vdc_to_wc (fildes:integer;vdcx,vdcy,vdcz:real;
var wcx,wcy,wcz:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

vdcx,vdcy,vdcz Virtual device coordinates, specified as real (float) numbers, to be transformed into world coordinates.

Output Parameters

wcx,wcy,wcz World coordinates.

Discussion

This function is now implemented by the *transform_point* routine. It is suggested that *trasnform_point* be called directly.

Calling this routine is equivalent to calling *transform_point* with its **mode** parameter set to VDC_TO_MC. If no modelling matrices have been pushed on the matrix stack, modelling and world coordinates are identical.

DEFAULT

2-dimensional multiplication.

SEE ALSO

transform_point(3G), pop_matrix(3G), push_matrix(3G), gopen(3G), wc_to_vdc(3G).

NAME

vertex_format – set vertex list format for polygons and polylines

SYNOPSIS

C Syntax:

void vertex_format(fildes,coord,use,rgb,normals,order);
int fildes,coord,use,rgb,normals,order;

FORTRAN77 Syntax:

subroutine vertex_format(fildes,coord,use,rgb,normals,order)
integer *4 fildes,coord,use,rgb,normals,order

Pascal Syntax:

procedure vertex_format(fildes,coord,use,rgb,normals,order:integer);

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

coord Set the number of extra coordinates supplied with each polygon, polyline and polymarker vertex. Must be greater than or equal to *use*.

use Sets the number of extra coordinates to be used for color determination. Allowable values are 0, 1, 3, or 6.

rgb Specifies if and where red, green and blue values are contained in each coordinate or if extra parameters are implied to be normals. Allowable values are 0, 1, or 4.

Valid combinations (brackets [] mark optional data; AppDep = application dependent use):

	no color coordinates	intensity	rgb	offset rgb	normal	rgb normal	normal rgb
coord	>=0	>=1	>=3	>=6	>=3	>=6	>=6
use	0	1	3	3	3	6	6
rgb	0	0	1	4	0	1	4
Vertex Data Format	x	x	x	x	x	x	x
	y	y	y	y	y	y	y
	[z]	[z]	[z]	z	z	z	z
	[AppDep]	intensity	r	AppDep	nx	r	nx
	[AppDep]	[AppDep]	g	AppDep	ny	g	ny
	[AppDep]	[AppDep]	b	AppDep	nz	b	nz
	[flag]	[AppDep]	[AppDep]	r	[AppDep]	nx	r
		[flag]	[AppDep]	g	[AppDep]	ny	g
			[AppDep]	b	[AppDep]	nz	b
			[flag]	[AppDep]	[flag]	[AppDep]	[AppDep]
				[AppDep]		[AppDep]	[AppDep]
				[AppDep]		[AppDep]	[AppDep]
				[flag]		[flag]	[flag]

normals If TRUE (1), it is assumed that the first entry in the vertex list of each **polygon3d** call is a normal to that polygon.

order If **order** is CLOCKWISE (0), polygon vertices are assumed to be arranged in a clockwise format. If **order** is COUNTER_CLOCKWISE (1), vertices are assumed to be counter-clockwise. The clockwise/counter-clockwise direction is defined in terms of viewing from the outside of the object, looking in from a left-handed coordinate system.

If **order** is ORed with UNIT_NORMALS, then all geometric and vertex normal vectors are assumed to be normalized in modelling coordinates.

Discussion

The **coord** parameter determines how many extra coordinates Starbase accepts for each vertex in non-integer interface *polyline*, *polymarker*, *polygon*, and *partial_polygon* procedures (extra coordinates are those coordinates that are in addition to x/y, x/y/z, x/y/flag, or x/y/z/flag). Vertex format has no effect on *intpolyline2d*, *intpolymarker2d*, *intpolygon2d*, *intpartial_polygon2d*, *intpolycircle*, and *intpolyrectangle*. Setting **coord** to a value larger than **use** provides a means for placing application-dependent data or color-determining data in the vertex list.

By setting **use** to:

0 All extra coordinates are ignored. Polylines are drawn in the current line color.

- With shading off, polygons are filled with the current fill color.
- With shading on, polygons are filled with the current fill color modified by the light source equations given in *surface_model*. This value produces constant (sometimes called faceted) shading.

1 The **coord** parameter must be 1 or greater and either CMAP_MONOTONIC or CMAP_FULL must have been specified in *shade_mode*. For polylines, an intensity is given for each vertex, and the intensity times the line color is linearly interpolated between vertices.

- With shading off, polygons use an intensity value per vertex to linearly interpolate the intensity times the fill color in the filled area.
- With shading on, the color of a polygon vertex is fill color multiplied by the intensity value which is then modified by the light source equations. The color at each pixel is linearly interpolated from the vertex values.

3 The **coord** parameter must be 3 or greater and either CMAP_MONOTONIC or CMAP_FULL must have been specified with *shade_mode*. For polylines a red, green, and blue value is given for each vertex and the colors are linearly interpolated between vertices. Normals per vertex are only used for filled polygons.

- With shading off, polygons use a red, green and blue value per vertex to linearly interpolate the colors in the filled area.
- With shading on and *rgb* equal one (1), the color of a polygon vertex is the red, green and blue color triplet located right after the x,y[,z] coordinates and modified by the light source equations. The color at each pixel is linearly interpolated from the vertex values.
- With shading on and *rgb* equal zero (0), the color of a polygon vertex is the fill color modified by the light source equations using the 3 extra coordinates located right after the x,y[,z] coordinates as the x, y, z normal to the vertex. The color at each pixel is linearly interpolated from the vertex values to produce smooth (Gouraud) shading.

- With shading on and *rgb* equal four (4), the color of a polygon vertex is the red, green and blue color triplet, starting in the fourth value after the z coordinate, modified by the light source equations. The color at each pixel is linearly interpolated from the vertex values.

6 If nonzero, the **coord** parameter must be 6 or greater and either CMAP_MONOTONIC or CMAP_FULL must have been specified with *shade_mode*. For polylines a red, green, blue, x_normal, y_normal and z_normal value is given for each vertex and the colors are linearly interpolated between vertices. Note that if **use** is 6, the first three extra coordinates will always be treated as RGB values for polylines, even if **rgb** is four (4). Normals per vertex are only used for filled polygons; they are ignored in polylines.

- With shading off, polygons use a red, green and blue value per vertex to linearly interpolate the colors in the filled area. These RGB values are assumed to be the first three extra coordinates.
- With shading on and *rgb* equal one (1), the color of a polygon vertex is the red, green and blue color triplet, following the z coordinate value, modified by the light source equations. The color at each pixel is linearly interpolated from the vertex values and specular highlights are added from the three extra coordinates following the blue color value which are the x,y,z normal to the vertex.
- With shading on and *rgb* equal four (4), the color of a polygon vertex is the red, green and blue color triplet, starting at the fourth value after the z coordinate value, modified by the light source equations. The color at each pixel is linearly interpolated from the vertex values and specular highlights are added from the three extra coordinates following the z coordinate value which are the x,y,z normal to the vertex.

If the **coord** parameter is nonzero, move/draw flags in the vertex data are found after all extra coordinates (used and not used).

Note that the color determinations described above for polygons apply only to the fill colors used. Polygon edges are always drawn using the current perimeter color.

When *normals* is FALSE (0), the light source equations either generate a normal to the polygon (by taking the cross product of the two vectors described by the first 3 vertices of a polygon list of vertices) or use the normals supplied with each vertex. If backfacing cull is on then a normal will always be generated. Note that since the normal is calculated using the first three polygon vertices, the vertices must neither be colinear nor very close to each other. Also, the calculated normal may lead to unusual results if the vertices form a concave section of the polygon or if the polygon is non-planar. During partial polygon calls, the normal calculations use the first three vertices of the first partial polygon. There must be at least three vertices in the first partial/full polygon or the calculations will be incorrect.

When *normals* is TRUE (1), the light source equations will either use the normal supplied at the beginning of each call to *polygon3d* or use the normal-per-vertex if supplied. If backfacing cull is on, the normal supplied at the beginning of the polygon is used to determine whether to render the polygon.

Normals only occur in the vertex list of *polygon3d* calls. They are not expected in *polygon2d*, *intpolygon2d*, *partial_polygon2d*, *intpartial_polygon2d*, or *partial_polygon3d* calls.

The **order** parameter is needed when polygon normals are generated. If an object is made of polygons, the vertices of all the polygons must be given in either a clockwise or counterclockwise direction in order for light sources and backfacing cull to work correctly.

When using a right-handed coordinate system, the sense of CLOCKWISE and COUNTER_CLOCKWISE are reversed, meaning that direction is determined from the inside of an object looking out. A right-handed coordinate system is obtainable by pushing an identity matrix onto the matrix stack with the z-scale term set to −1.

Providing normalized surface normals within the data and ORing the UNIT_NORMALS flag into the **order** parameter will cause the Starbase program to execute more quickly, since the normalization calculation will not be required during rendering. A normalized vector is a vector with a length of one unit.

Spline surface generation is also affected by the current *vertex_format*. If **coord** = 3, **use** = 3, and **rgb** = 0 (normals per vertex), spline surface generation produces normals per vertex with smooth (Gouraud) shading. In all other cases, constant (or faceted) shading is used for spline surfaces. Spline surfaces also use the current *interior_style* which must be set to INT_SOLID before any shading can occur.

For performance reasons, rgb data in vertex lists is not checked for validity. These values should range from 0.0 to 1.0 for floating point calls (for example, *polygon*); for calls using DC values (for example, *dcpolygon*), the values should range from 0 to 32,000. If they are not in the correct range, device-dependent behavior will result.

Performance is increased when **coord** is equal to **use** .

DEFAULTS

coord = 0, **use** = 0, **rgb** = 0, **normals** = FALSE (0), **order** = COUNTER_CLOCKWISE (1).

SEE ALSO

backface_control(3G), hidden_surface(3G), shade_mode(3G), spline(3G), surface_model(3G), *Starbase Graphics Techniques.*

NAME

view_camera – define 3d viewing transformation matrix using camera model

SYNOPSIS

C Syntax:

```
void view_camera(fildes,camera);
int fildes;
camera_arg *camera;
```

FORTRAN77 Syntax:

```
subroutine view_camera(fildes,camera)
integer*4 fildes
real camera(13)
```

Pascal Syntax:

```
procedure view_camera(fildes:integer;var camera:camera_arg);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

camera Structure, record, or array that defines the camera model orientation and contains the following individual elements:

camera position – 3d world coordinates of camera.

C and Pascal: camera.camx, camera.camy, camera.camz
FORTRAN: camera(CAM_CAMX), camera(CAM_CAMY), camera(CAM_CAMZ)

reference position – 3d world coordinates of center of interest.

C and Pascal: camera.refx, camera.refy, camera.refz
FORTRAN: camera(CAM_REFX), camera(CAM_REFY), camera(CAM_REFZ)

field of view – angular field of view in degrees

C and Pascal: camera.field_of_view
FORTRAN: camera(CAM_FIELD_OV)

clipping planes – front and back clipping planes in world coordinates. If both planes are equal, the front plane is set to be very close to the camera and the back plane is set to be far away. If the front is set behind the camera position, it is changed to be just in front of the camera.

C and Pascal: camera.front, camera.back
FORTRAN: camera(CAM_FRONT), camera(CAM_BACK)

camera up vector – 3d world coordinates vector from camera bottom to camera top. This vector is projected onto the plane that is normal to the vector from the camera to the reference position.

C and Pascal: camera.upx, camera.upy, camera.upz
FORTRAN: camera(CAM_UPX), camera(CAM_UPY), camera(CAM_UPZ)

camera projection type – either CAM_PERSPECTIVE or CAM_PARALLEL

C and Pascal: camera.projection
FORTRAN: camera(CAM_PROJECTION)

Discussion

This function is used to define a new **viewing transformation** matrix with a simple 3-dimensional camera model. This camera view is mapped into the current *view_port* in a non-distorted fashion. Subsequent calls to *view_port* recalculate the viewing transformation matrix using the camera view established by this call until another Starbase function establishes a new viewing transformation.

The **viewing transformation** matrix is the world-to-device coordinate transformation, and should include all user defined non-linear transformations such as perspective. Modelling transformations that affect objects and not the view, such as object translations, rotations, uniform scaling, etc., should appear in modelling matrices pushed on the matrix stack.

If this function is called when the device is not in MODEL_XFORM mode, the matrix stack is flushed since old transformations based on the viewing transformation are invalidated. If this function is called when the device is in the MODEL_XFORM mode, the viewing matrix (at the bottom of the matrix stack) is changed without changing any of the matrices in the rest of the stack.

A viewing transformation matrix based on the camera data in the structure/record/array passed in to view_camera, is calculated. The camera model is really quite simple; the transformation is set up to look from the camera position, to the reference position. The camera geometry is defined by the angular field of view of the camera in degrees (typically 60 degrees). In parallel projections, this angle is only accurate in the plane of the reference position. Changing this value is akin to zooming in or out with a real camera. The camera front and back are the distances in world coordinates to the front and back clipping planes, relative to the reference position, along the direction vector (from camera to reference position). These values reset the values that were in effect from *clip_depth* or the last *view_camera*. They also reset the values of the *depth_cue_range*. The camera up vector determines the orientation of the camera, i.e., which direction the top of the camera points. The camera itself is aimed in the direction established by the vector from the camera position to the reference position. Finally the type of projection used by the camera is determined by the camera projection type.

If camera.projection = CAM_PERSPECTIVE, a perspective projection is used; if camera.projection = CAM_PARALLEL, a parallel projection is used. The *view_camera* function assumes a left-handed world coordinate system. If this is not the case, then simply call the *view_matrix3d* function right after calling the *view_camera* function with a matrix that is identity except that the z-scale term is -1, and the mode set to PRE_CONCAT_VW.

view_camera also sets the viewpoint used for specular light sources. If the viewpoint is currently set to be positional, the viewpoint is set to be the camera position. If the viewpoint is directional, the viewpoint vector is set to be the vector from the reference position to the camera position. If the projection is perspective (camera.projection = CAM_PERSPECTIVE) and the viewpoint is directional, this function sets the positional viewpoint first, then sets the directional viewpoint as specified above. This ensures that back culling works correctly (see *hidden_surface*).

A simple example further explains: To view a unit cube centered at −5,0,−10 from the positive z direction(left-handed coordinates) and slightly above in the y direction, the following call to view_camera would work:

```
camera_arg camera;
camera.back=2.0;    camera.front=-2.0;    camera.upx=0.0;
camera.camx=-5.0;   camera.refx=-5.0;     camera.upy=1.0;
camera.camy=1.0;    camera.refy=0.0;      camera.upz=0.0;
camera.camz=-7.0;   camera.refz=-10.0;
camera.field_of_view=60.0;
camera.projection = CAM_PERSPECTIVE;
view_camera(fildes,&camera);
draw_cube();
```

The coordinate systems used by Starbase can be conceptually defined as follows:

- User points are assumed to be defined in modelling coordinates. These points are transformed into world coordinates using a matrix called the **modelling transformation matrix** (if any matrices have been pushed on the stack). World coordinates are used to perform any rendering calculations needed.
- Next, the world coordinates are transformed to device coordinates by the **viewing transformation matrix**, which is usually the concatenation of user-defined viewing transformations and the vdc-to-device transformation matrix. When a device is opened, the **viewing transformation matrix** is simply the vdc-to-device coordinate transformation matrix.

Use matrix stack manipulation functions to place modelling transformation matrices on the matrix stack. Use *view_camera*, *view_matrix*, *view_volume*, or *view_window* to set viewing transformations.

- If a graphics device has been opened in MODEL_XFORM mode, the modelling transformation cannot be combined with the viewing transformation because special rendering calculations such as shading may be needed after the modelling-to-world-coordinate transformation. Therefore, matrices pushed on the stack are left undisturbed and all transformations from modelling coordinates to device coordinates are processed in two steps: modelling to world coordinates, followed by world to device coordinates.
- If a graphics device is not in MODEL_XFORM mode, the modelling and viewing transformations can be combined. Thus, the current viewing transformation matrix is post-concatenated to modelling matrices placed on the matrix stack, and subsequent output primitives are transformed, using only the top matrix on the matrix stack.

DEFAULTS

After *gopen* or a call to *flush_matrices*, the current viewing transformation matrix is the vdc-to-device units transformation matrix.

SEE ALSO

flush_matrices(3G), gopen(3G), hidden_surface(3G), vdc_extent(3G), view_matrix(3G), view_port(3G), view_window(3G), viewpoint(3G), *Starbase Graphics Techniques.*

NAME

intview_matrix2d, view_matrix2d, view_matrix3d – define a viewing transformation matrix

SYNOPSIS

C Syntax:

```
void intview_matrix2d(fildes,xform2,radix,usage,raw);
int fildes,usage,xform2[3][2],radix,usage,raw;

void view_matrix2d(fildes,xform2,usage);
int fildes,usage;
float xform2[3][2];

void view_matrix3d(fildes,xform3,usage);
int fildes,usage;
float xform3[4][4];
```

FORTRAN77 Syntax:

(See Language Dependencies below)

```
subroutine intview_matrix2d(fildes,xform2,radix,usage,raw)
integer/(**4 fildes,xform2(2,3),radix,usage,raw

subroutine view_matrix2d(fildes,xform2,usage)
integer/(**4 fildes,usage
real xform2(2,3)

subroutine view_matrix3d(fildes,xform3,usage)
integer/(**4 fildes,usage
real xform3(4,4)
```

Pascal Syntax:

```
type
    int2d_xform=array[1..3][1..2] of integer;
    two_d_xform=array[1..3][1..2] of real;
    three_d_xform=array[1..4][1..4] of real;

procedure intview_matrix2d(fildes:integer;var xform2:int2d_xform;
    radix,usage,raw:integer);

procedure view_matrix2d(fildes:integer;var xform2:two_d_xform;usage:integer);

procedure view_matrix3d(fildes:integer;var xform2:three_d_xform;usage:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

xform2 A 3x2 (2-Dimensional) matrix.

xform3 A 4x4 (3-Dimensional) matrix.

radix is the radix factor for the 3x2 integer matrix.

usage If set to REPLACE_VW, the old viewing transformation is discarded and the new viewing transformation is the user-supplied matrix concatenated with the vdc-to-device units transformation matrix.

If set to PRE_CONCAT_VW, the user-supplied matrix is pre-concatenated with the current viewing transformation matrix. This result becomes the new viewing transformation matrix.

If set to POST_CONCAT_VW, the user-supplied matrix is post-concatenated with the current viewing transformation matrix. This result becomes the new viewing transformation matrix.

raw If set to TRUE(1), integer matrices will be in internal (raw) format (i.e. translation values are not scaled).
If set to FALSE(0), all integer matrix values are scaled.

Discussion

View_matrix explicitly define a new **viewing transformation matrix**. The functions *intview_window*, *intview_port*, *view_window*, *view_volume*, *view_camera*, and *view_port* can also be used to set this matrix using simple 2d window/viewport, 3d volume/viewport, and 3d camera models. If this function is called when one of the above viewing models is in effect, subsequent calls to *view_port* or *intview_port* will not result in a change to the viewing transformation matrix. The **viewing transformation matrix** is the world-to-device-coordinate transformation, and should include all user-defined non-linear transformations such as perspective. Modelling transformations that affect objects but not the view (such as object translations, rotations, uniform scaling, etc.) should appear in modelling matrices pushed on the matrix stack.

When *view_matrix* is called, the matrix stack is flushed because old transformations based on the viewing transformation are no longer valid.

If this function is called when the device is not in MODEL_XFORM mode, the matrix stack is flushed because old transformations based on the viewing transformation are no longer valid. If this function is called when the device is in MODEL_XFORM mode, the viewing matrix (at the bottom of the matrix stack) is changed without changing any of the matrices in the rest of the stack.

The new viewing transformation matrix is formed according to the **usage** parameter:

- If **usage** is set to REPLACE_VW, the user-supplied matrix **xform2** or **xform3** is concatenated with the current vdc-to-device units transformation matrix, resulting in a new viewing transformation matrix (the old matrix is discarded). This concatenation step can be avoided by setting vdc equal to device coordinates (see *vdc_extent* and *intvdc_extent*.)
- If **usage** is PRE_CONCAT_VW, the specified matrix is pre-concatenated with the current viewing transformation matrix and the result is the new viewing transformation matrix.
- If **usage** is POST_CONCAT_VW, the new viewing transformation matrix is formed from the post-concatenation of the user supplied matrix with the current viewing transformation matrix.

Post-concatenation should be used very carefully. It is not device-independent because the post concatenation occurs after the vdc-to-device coordinate transformation. The user of this feature should determine the range of device coordinates by using *inquire_sizes*.

Starbase maintains all internal floating point matrices in 3-dimensional (4x4) form, so *view_matrix2d* must expand the 2d matrix **xform2**.

Since the vdc-to-device units transformation matrix is usually post-concatenated to the end of this matrix, care should be taken when performing perspective transformations. A perspective model where the eye is at origin of perspective space is recommended. Any other model can easily be modified to this perspective model by a single translation step.

The coordinate systems used by Starbase can be conceptually defined as follows:

- User points are assumed to be defined in modelling coordinates. These points are transformed into world coordinates using a matrix called the **modelling transformation matrix** (if any matrices have been pushed on the stack). World coordinates are used to perform any rendering calculations needed.

- World coordinates are transformed to device coordinates by the *viewing transformation matrix*, which is usually the concatenation of user-defined viewing transformations and the vdc-to-device transformation matrix. When a device is opened, the *viewing transformation matrix* is simply the vdc-to-device coordinate transformation matrix. *View_window*, *view_port*, *view_matrix*, *view_volume*, *view_camera*, *intview_window*, *intview_port*, and *intview_matrix2d* can be used to define further viewing transformations.

Use matrix stack manipulation functions to place modelling transformation matrices on the matrix stack. Use *view_matrix* to set viewing transformations.

- If a graphics device has been opened in MODEL_XFORM mode, the modelling transformation cannot be combined with the viewing transformation because special rendering calculations such as shading may be needed after the modelling-to-world-coordinate transformation. Therefore, matrices pushed on the stack are left undisturbed and all transformations from modelling coordinates to device coordinates are processed in two steps: modelling to world coordinates, followed by world to device coordinates.
- If a graphics device is not in MODEL_XFORM mode, the modelling and viewing transformations can be combined. Thus, the current viewing transformation matrix is post-concatenated to modelling matrices placed on the matrix stack, and subsequent output primitives are transformed, using only the top matrix on the matrix stack.

When using *view_matrix* to generate a 3-dimensional perspective view, the *viewpoint* function should be called to establish the observer's eye position so that back-cull calculations will be correct. In addition, *clip_depth* should be called to determine front and back clip planes, and *depth_cue_range* should be called to set the range of the depth cue.

Integer transformation matrices are scaled to allow a fractional portion for rotating objects. The radix factor indicates the number of bits to the right of the decimal point. Legal limits are 0 to 30. Once a coordinate has been transformed, it is divided by 2****radix** to return to an integer value.

When using raw mode with an integer matrix positions (3,1) and (3,2) have an implied radix factor of 0 (no scaling). This allows large translation ranges along with accurate rotations.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode Floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of Integer operations, Floating point operations and common operations see the *starbase.3g* manual page.

LANGUAGE DEPENDENCIES

FORTRAN77:

FORTRAN requires a transposition of array rows and columns due to the manner in which FORTRAN77 stores arrays.

DEFAULTS

After *gopen* or a call to *flush_matrices*, the current viewing transformation matrix is the vdc-to-device units transformation matrix.

SEE ALSO

clip_depth(3G), depth_cue_range(3G), flush_matrices(3G), gopen(3G), inquire_sizes(3G), view_window(3G), view_volume(3G), view_camera(3G), viewpoint(3G), view_port(3G), vdc_extent(3G), *Starbase Graphics Techniques*.

NAME

intview_port, view_port – define mapping area on view_surface for subsequent view_windows, view_volumes, and view_cameras

SYNOPSIS

C Syntax:

```
void intview_port(fildes,x1,y1,x2,y2);
int fildes,x1,y1,x2,y2;

void view_port(fildes,x1,y1,x2,y2);
int fildes;
float x1,y1,x2,y2;
```

FORTRAN77 Syntax:

```
subroutine intview_port(fildes,x1,y1,x2,y2);
integer*4 fildes,x1,y1,x2,y2;

subroutine view_port(fildes,x1,y1,x2,y2);
integer*4 fildes
real x1,y1,x2,y2;
```

Pascal Syntax:

```
procedure intview_port(fildes,x1,y1,x2,y2:integer);

procedure view_port(fildes:integer; real x1,y1,x2,y2);
```

DESCRIPTION

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

x1,y1,x2,y2 Boundaries of the new viewport specified in virtual device coordinates (vdcs).

Discussion

This function defines a new viewport on the virtual device that *intview_window*, *view_window*, *view_volume*, and *view_camera* functions map to. It also sets the range of vdc space that is to be zbuffered when hidden surface removal is active.

When using *view_window* or *view_volume*, the entire window/volume (in world coordinates) maps to the entire viewport (in vdcs). This causes distortion if the aspect ratios of the two are not the same or if mapping mode is set to DISTORT. When using *view_camera*, the camera view will appear within the bounds of the viewport with no distortion.

After changing the viewport range, this function tries to re-calculate the viewing transformation matrix if the last viewing transformation matrix was set with *intview_window*, *view_window*, *view_volume*, or *view_camera*. If the last viewing transformation matrix was set with *view_matrix*, or if the viewing transformation matrix is the vdc-to-device matrix, *view_port* does not try to recalculate the viewing transformation matrix. The viewing transformation matrix is the vdc-to-device matrix if no viewing function (*view_window*, *view_volume*, *view_camera*, or *view_matrix*) has been called since *gopen* time or since the last call to *flush_matrices*.

This viewport is truncated to the current *vdc_extent* if it exceeds the range of vdc.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

DEFAULTS

After *gopen*, the default viewport is the same as the default *vdc_extent*.

SEE ALSO

gopen(3G), hidden_surface(3G), inquire_sizes(3G), view_camera(3G), view_window(3G),

view_volume(3G), view_matrix(3G), vdc_extent(3G), zbuffer_switch(3G), *Starbase Graphics Techniques*.

NAME

view_volume – define 3d viewing transformation matrix using a volume/viewport model

SYNOPSIS

C Syntax:

```
void view_volume(fildes,x1,y1,z1,x2,y2,z2);
int fildes;
float x1,y1,z1,x2,y2,z2;
```

FORTRAN77 Syntax:

```
subroutine view_volume(fildes,x1,y1,z1,x2,y2,z2);
integer*4 fildes
real x1,y1,z1,x2,y2,z2;
```

Pascal Syntax:

```
procedure view_volume(fildes:integer; real x1,y1,z1,x2,y2,z2);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

x1,y1,x2,y2,z1,z2
Boundaries of the new volume specified in world coordinates.

Discussion

This function is used to define a new **viewing transformation** matrix with a simple volume/viewport model. The viewing transformation matrix is the world- to device-coordinate transformation, and should include all user-defined non-linear transformations such as perspective. Modelling transformations that affect objects but not the view, such as object translations, rotations, uniform scaling, etc., should appear in modelling matrices pushed on the matrix stack.

If this function is called when the device is not in MODEL_XFORM mode, the matrix stack is flushed since old transformations based on the viewing transformation are invalidated. If this function is called when the device is in the MODEL_XFORM mode, the viewing matrix (at the bottom of the matrix stack) is changed without changing any of the matrices in the rest of the stack.

The new viewing transformation matrix is computed, based on the new volume bounds and the current viewport definition. The entire x and y range of the view_volume (in world coordinates) maps to the entire viewport (in vdcs) in a distorted fashion if the aspect ratios of the two are not the same. The entire range of the *view_volume* in the z dimension maps to the entire z range of the vdc_extent. Subsequent calls to *view_port* recalculate the viewing transformation matrix using the view volume established by this call and the new viewport until another starbase function establishes a new viewing transformation. The *clip_depth* and *depth_cue_range* are also reset to coincide with the current *vdc_extent* bounds in the z direction.

The coordinate systems used by Starbase can be conceptually defined as follows:

- User points are assumed to be defined in modelling coordinates. These points are transformed into world coordinates using a matrix called the **modelling transformation matrix** (if any matrices have been pushed on the stack). World coordinates are used to perform any rendering calculations needed.
- Next, the world coordinates are transformed to device coordinates by the **viewing transformation matrix**, which is usually the concatenation of user-defined viewing transformations and the vdc-to-device transformation matrix. When a device is opened, the **viewing transformation matrix** is simply the vdc-to-device coordinate transformation matrix.

Use matrix stack manipulation functions to place modelling transformation matrices on the matrix stack. Use *view_camera*, *view_matrix*, *view_port*, *view_volume*, or *view_window* to set viewing transformations.

- If a graphics device has been opened in MODEL_XFORM mode, the modelling transformation cannot be combined with the viewing transformation because special rendering calculations such as shading may be needed after the modelling-to-world-coordinate transformation. Therefore, matrices pushed on the stack are left undisturbed and all transformations from modelling coordinates to device coordinates are processed in two steps: modelling to world coordinates, followed by world to device coordinates.
- If a graphics device is not in MODEL_XFORM mode, the modelling and viewing transformations can be combined. Thus, the current viewing transformation matrix is post-concatenated to modelling matrices placed on the matrix stack, and subsequent output primitives are transformed, using only the top matrix on the matrix stack.

DEFAULTS

After *gopen* or a call to *flush_matrices*, the current viewing transformation matrix is the vdc-to-device units transformation matrix.

SEE ALSO

flush_matrices(3G), gopen(3G), inquire_sizes(3G), vdc_extent(3G), view_camera(3G), view_matrix(3G), view_port(3G), view_window(3G), *Starbase Graphics Techniques*.

NAME

intview_window, view_window – define a 2d viewing transformation matrix using a window/viewport model

SYNOPSIS

C Syntax:

```
void intview_window(fildes,x1,y1,x2,y2);
int fildes,x1,y1,x2,y2;

void view_window(fildes,x1,y1,x2,y2);
int fildes;
float x1,y1,x2,y2;
```

FORTRAN77 Syntax:

```
subroutine intview_window(fildes,x1,y1,x2,y2);
integer*4 fildes,x1,y1,x2,y2;

subroutine view_window(fildes,x1,y1,x2,y2);
integer*4 fildes
real x1,y1,x2,y2;
```

Pascal Syntax:

```
procedure intview_window(fildes,x1,y1,x2,y2:integer);

procedure view_window(fildes:integer; real x1,y1,x2,y2);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

x1,y1,x2,y2 Boundaries of the new window specified in world coordinates.

Discussion

This function is used to define a new *viewing transformation matrix* with a simple 2-dimensional window/viewport model. The *viewing transformation matrix* is the world to device coordinate transformation, and should include all user defined non-linear transformations such as perspective. Modelling transformations that affect objects but not the view, such as object translations, rotations, uniform scaling, etc., should appear in modelling matrices pushed on the matrix stack.

When this function is called the matrix stack is flushed because old transformations based on the viewing transformation are no longer valid.

If this function is called when the device is not in MODEL_XFORM mode, the matrix stack is flushed because old transformations based on the viewing transformation are no longer valid. If this function is called when the device is in the MODEL_XFORM mode, the viewing matrix (at the bottom of the matrix stack) is changed without changing any of the matrices in the rest of the stack.

The viewing transformation matrix is based on the new window bounds and the current viewport definition. The entire window (in world coordinates) maps to the entire viewport (in vdcs) in a distorted fashion if the aspect ratios of the two are not the same. Subsequent calls to *view_port* recalculate the viewing transformation matrix using the window established by this call until another starbase function establishes a new viewing transformation matrix.

The coordinate systems used by Starbase can be conceptually defined as follows:

- User points are assumed to be defined in modelling coordinates. These points are transformed into world coordinates using a matrix called the *modelling transformation matrix* (if any matrices have been pushed on the stack). World coordinates are used to perform any rendering calculations needed.

- World coordinates are transformed to device coordinates by the *viewing transformation matrix*, which is usually the concatenation of user-defined viewing transformations and the vdc-to-device transformation matrix. When a device is opened the *viewing transformation matrix* is simply the vdc-to-device coordinate transformation matrix. *View_window*, *view_matrix*, *view_port*, *view_volume*, *view_camera*, *intview_window*, *intview_port*, and *intview_matrix2d* can be used to define further viewing transformations.

Use matrix stack manipulation functions to place modelling transformation matrices on the matrix stack. Use *view_camera*, *view_matrix*, *view_port*, *view_volume*, or *view_window* to set viewing transformations.

- If a graphics device has been opened in MODEL_XFORM mode, the modelling transformation cannot be combined with the viewing transformation because special rendering calculations such as shading may be needed after the modelling-to-world-coordinate transformation. Therefore, matrices pushed on the stack are left undisturbed and all transformations from modelling coordinates to device coordinates are processed in two steps: modelling to world coordinates, followed by world to device coordinates.
- If a graphics device is not in MODEL_XFORM mode, the modelling and viewing transformations can be combined. Thus, the current viewing transformation matrix is post-concatenated to modelling matrices placed on the matrix stack, and subsequent output primitives are transformed, using only the top matrix on the matrix stack.

Integer operations are only available when using the INT_XFORM gopen mode. When in INT_XFORM mode, floating point operations are not available for that fildes. Floating point operations are the default, or can be specified with FLOAT_XFORM mode. For a list of integer operations, floating point operations and common operations see the *starbase.3g* manual page.

DEFAULTS

After *gopen* or a call to *flush_matrices*, the current viewing transformation matrix is the vdc-to-device units transformation matrix.

SEE ALSO

flush_matrices(3G), gopen(3G), inquire_sizes(3G), vdc_extent(3G), view_camera(3G), view_matrix(3G), view_port(3G), view_volume(3G), *Starbase Graphics Techniques*.

NAME

viewpoint – define eye position in world coordinates for use when calculating specular reflections and back-culling polygons.

SYNOPSIS

C Syntax:

```
void viewpoint(fildes,type,x,y,z);
int fildes,type;
float x,y,z;
```

FORTRAN77 Syntax:

```
subroutine viewpoint(fildes,type,x,y,z)
integer*4 fildes,type
real x,y,z
```

Pascal Syntax:

```
procedure viewpoint(fildes, type:integer; x, y, z:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

type Selects DIRECTIONAL or POSITIONAL viewpoint interpretation.

x,y,z Coordinate position of the viewpoint or a vector pointing to the eye point

Discussion

Viewpoint specifies the location of the viewpoint for use when calculating specular reflections. Light source equations require a known angle between the viewer and the reflected ray in order to determine the specular reflection.

The **type** parameter can be one of the following:

DIRECTIONAL Viewpoint at infinity. The **x,y,z** parameters specify a vector anchored at the origin and directed toward the viewpoint. Note that this vector does not need to be normalized to unit length.

POSITIONAL Viewpoint close to viewing area. The **x,y,z** parameters specify the position of the viewpoint in world coordinates.

Viewpoint should not be confused with *view_camera* which allows the user to define a 3-dimensional viewing transformation using a camera model. The *view_camera* function calls *viewpoint* to guarantee that the eyepoint used for specular reflection calculations is consistent with the camera position.

When set in positional mode, the eyepoint is also used for back-culling calculations when the viewing transformation matrix is a perspective transformation (see *hidden_surface*). This is required whenever using *view_matrix3d* to do perspective viewing. When using *view_camera*, the viewpoint position set by *view_camera* is the camera position.

DEFAULTS

Viewpoint at infinity along the -z axis.

SEE ALSO

hidden_surface(3G), surface_model(3G), shade_mode(3G), view_camera (3G), *Starbase Graphics Techniques.*

NAME

wc_to_vdc – transform world coordinate point into virtual device coordinate point using current transformation matrix

SYNOPSIS

C Syntax:

```
void wc_to_vdc(fildes,wcx,wcy,wcz,vdcx,vdcy,vdcz);
int fildes;
float wcx,wcy,wcz,*vdcx,*vdcy,*vdcz;
```

FORTRAN77 Syntax:

```
subroutine wc_to_vdc(fildes,wcx,wcy,wcz,vdcx,vdcy,vdcz)
integer*4 fildes
real wcx,wcy,wcx,vdcx,vdcy,vdcz
```

Pascal Syntax:

```
procedure wc_to_vdc(fildes:integer;wcx,wcy,wcz:real;
var vdcx,vdcy,vdcz:real);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphic device is opened.

wcx,wcy,wcz Define a point in world coordinate space.

Output Parameters

vdcx,vdcy,vdcz

Transformed point in virtual device coordinate space.

Discussion

This function is now implemented by the *transform_point* routine. It is suggested that *transform_point* be called directly.

This routine is equivalent to calling *transform_point* with its *mode* parameter set to MC_TO_VDC. If no modelling matrices have been pushed on the matrix stack, modelling and world coordinates are identical.

SEE ALSO

gopen(3G), pop_matrix(3G), push_matrix(3G), transform_point(3G), wc_to_vdc(3G), view_camera(3G), view_matrix(3G).

NAME

write_enable – select modifiable planes of frame buffer device

SYNOPSIS

C Syntax:

```
void write_enable(fildes,plane_mask);
int fildes;
int plane_mask;
```

FORTRAN77 Syntax:

```
subroutine write_enable(fildes,plane_mask)
integer*4 fildes,plane_mask
```

Pascal Syntax:

```
procedure write_enable(fildes,plane_mask:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

plane_mask

Mask used to determine which planes are enabled for modification. Each bit is interpreted as the plane enable of that respective plane; i.e., if the least significant bit is 0 then plane 1 is disabled. If the 2nd bit is 1, then plane 2 is enabled, etc.

Discussion

If a plane is disabled, writing into the frame buffer will not affect that plane. For example, if the **plane_mask** is 15 (hex 0F) and 170 (hex AA) is written into a location containing 85 (hex 55), that location will change to 90 (hex 5A).

All raster operations and output primitives are affected by *write_enable*.

For multi-bank graphics devices *plane_mask* is a byte quantity that affects the corresponding planes of all banks simultaneously. See *bank_switch* to enable a bank of eight planes for writing. Graphics devices that support video blending may treat the plane mask as a 24 bit quantity while blending is enabled.

Write_enable is not supported on certain devices.

Double-buffering with 12 planes or with fewer than 6 planes uses *display_enable* and *write_enable* to select buffers (other values use *bank_switch*). During double buffering, *display_enable* and *write_enable* are applied to only the displayed and written planes as determined by double_buffering.

DEFAULTS

plane_mask = 255: All planes are enabled.

SEE ALSO

display_enable(3G), double_buffer(3G), *Starbase Device Driver Library Manual.*

NAME

zbuffer_switch – enable section of display surface for zbuffer hidden surface removal

SYNOPSIS

C Syntax:

```
void zbuffer_switch(fildes,pass);
int fildes,pass;
```

FORTRAN77 Syntax:

```
subroutine zbuffer_switch(fildes,pass)
integer *4 fildes,pass
```

Pascal Syntax:

```
procedure zbuffer_switch(fildes,pass:integer);
```

DESCRIPTION

Input Parameters

fildes Integer file descriptor returned by *gopen* when an I/O path to a graphics device is opened.

pass Number specifying which pass of the user data is to be expected.

Discussion

If multiple passes of the user data are needed to do complete hidden surface removal, the frame buffer is divided into vertical strips and rendered one at a time. The number of strips is determined by the *hidden_surface* procedure. When this procedure is called, the zbuffer is cleared and the clip limits are set to the intersection of the current clip rectangle and the rectangle defining the requested strip of frame buffer as defined by the **pass** parameter. Pass 1 renders the left-most strip and subsequent passes render strips left to right. This procedure should be called even when multiple passes are not necessary in order to clear out the zbuffer. This is usually done whenever *clear_view_surface* or *dbuffer_switch* is called.

Some devices which have dedicated zbuffers support clearing of the zbuffer *simultaneously* with the display. This can result in much better performance and should be used **instead** of *zbuffer_switch* whenever possible. To clear the zbuffer when *clear_view_surface* is called, *clear_control* must be called with CLEAR_ZBUFFER included in the **mode** parameter. While this mode is enabled *zbuffer_switch* will NOT clear the zbuffer. See *clear_control* for details.

DEFAULTS

pass 1 is expected.

SEE ALSO

hidden_surface(3G), clear_control(3G), *Starbase Graphics Techniques.*

NAME

bitmapfile – Starbase bitmap file

DESCRIPTION

A bitmap file created by Starbase for bitmap archival or printing has the following overall structure. The offsets to bitmap data and user data areas are recorded in fields of the header structure.

byte offset	contents
0	header structure (see below)
256	color map (if present)
arbitrary *	bitmap data
arbitrary **	user data (if present)

* depends upon size of the color map
** depends upon size of color map and bitmap data

Header

The 256 byte header block has the following structure:

```
struct bf_header
{
        char file_id[16];       /* "Bitmapfile" */
        int rev;               /* revision of file format */
        char device_id[16];     /* device or driver name
                                (e.g. hp300h) */
        int bm_loc;             /* offset in bytes to start of
                                bitmap information bytes
                                                measured from the start
                                                of the file */
        int eod_loc;            /* byte offset of last byte of
                                HP data in the file.
                                User data may be appended
                                                after this point. */
        int xstart,ystart;      /* upper left corner of source
                                in pixels (This is pixel 0,0
                                                in the file */
        int xlen,ylen;          /* x,y dimensions of bitmap
                                in pixels */
        int bm_mode;            /* format of bitmap data:
                                -1 -- pixel major full depth,
                                -2 -- plane major full depth,
                                >= 0 -- plane major, single plane
                                (source plane was this number.)
                                                (plane 0 = least significant.) */
        int depth;              /* number of bits stored per pixel.
                                This indicates how many planes
                                were stored. */
        int pixel_align;        /* alignment of pixel data in the file.
                                                Allowable values are 1 and 8.
                                1 means every bit is a pixel,
                                8 means every byte is a pixel. */
```

```
    int num_banks;        /* Number of banks. Only relevant
                          for pixel-major format. */
    int disp_en;              /* display enable mask of source */
    int cmap_mode;        /* color map mode:
                          0 = normal color mode.
                          1 = monotonic color mode.
                          4 = full color mode. */
    int csize;            /* Number of color map entries */
    int back_index;       /* Index value of background color
                          entry in the color map */
    char hp_extend[100];  /* Reserved for HP extensions */
    char user_extend[64]; /* Reserved for user extensions */
}
```

Color Map

The color map is stored as an array of float triples, representing red, green, and blue levels in the range 0.0 to 1.0. The color map starts at byte offset 256; the number of entries is indicated by the *csize* field in the header structure. This array is defined as:

```
float cmap[csize][3];
```

Bitmap Data

Bitmap data may be stored either in pixel-major or plane-major format. Single plane bitmaps are stored in plane-major format.

Pixel-major format

Pixel-major means that bits from several planes, all corresponding to one pixel, are stored in a single byte. When there are more than eight planes, the data is written in banks; all the data for the the least significant eight planes are written as the first bank, followed by the data for the next eight planes as the second bank, and so on up to the most significant planes in the last bank. Within a bank, consecutive bytes correspond to consecutive pixels. The last byte of the first row is followed immediately by the first byte of the second row, etc. Pixels are ordered first by x value from left (column 0) to right (column *xlen*-1), and second by y value from top (row 0) to bottom (row *ylen*-1). Banks are written without padding; the last byte of a bank is followed immediately by the first byte of the next bank. The storage required per bank in the file is thus *xlen* * *ylen* bytes.

Plane-major format

Plane-major means that all the data from one plane, for all pixels, are stored in adjacent positions in the file. This means that the bits from a single plane are packed, eight pixels per byte. Each pixel row begins on the next byte boundary. Pixels are ordered as for pixel-major data. Consequently, the most significant bit of the first byte corresponds to the leftmost pixel of the first row, and the least significant bit of the first byte corresponds to the eighth pixel. The storage required per plane is *((xlen + 7) div 8)* * *ylen* bytes.

Planes are padded only to the next byte boundary; when multiple planes are stored in a file, the last byte of each plane is followed immediately by the first byte of the next plane. Planes are stored in the file consecutively from least significant to most significant.

SEE ALSO

HP-UX Reference, pcltrans(1), lp(1M).

*Starbase Reference,*bitmap_to_file(3G), file_to_bitmap(3G), file_print(3G).

Starbase Graphics Techniques,"Storing and Printing Images".

Index

Win an HP Calculator!

Your comments and suggestions help us determine how well we meet your needs. **Returning this card with your name and address enters you into a quarterly drawing for an HP calculator***.

Starbase Reference

	Agree				**Disagree**
The manual is well organized.	○	○	○	○	○
It is easy to find information in the manual.	○	○	○	○	○
The manual explains features well.	○	○	○	○	○
The manual contains enough examples.	○	○	○	○	○
The examples are appropriate for my needs.	○	○	○	○	○
The manual covers enough topics.	○	○	○	○	○
Overall, the manual meets my expectations.	○	○	○	○	○

You have used this product:

__ Less than 1 week
__ Less than 1 month
__ Less than 1 year
__ 1 to 2 years
__ More than 2 years

fold ——

Please write additional comments, particularly if you disagree with a statement above. Use additional pages if you wish. The more specific your comments, the more useful they are to us.

Comments: _____

*Offer expires 7/1/1991. (Manual: 98592-90065 E0989)

Please Tape Here

Please print or type your name and address.

Name: _____

Company: _____

Address: _____

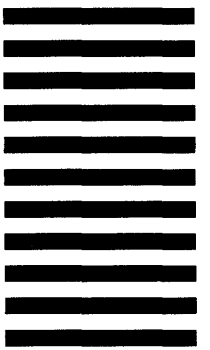
City, State, Zip: _____

Telephone: _____

Additional Comments: _____

Starbase Reference
HP Part Number 98592-90065
E0989

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



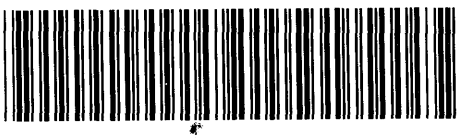
BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 37 LOVELAND, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

Hewlett-Packard Company
Attn: Learning Products Center
3404 East Harmony Road
Fort Collins, Colorado 80525-9988

HP Part Number
98592-90065
Microfiche No. 98592-99065
Printed in U.S.A. E0989



For Internal Use Only