



505 Wall Street • Chico, CA 95928 • ☎ 530-894-8400 📠 530-894-9069
www.scitechsoft.com • ftp.scitechsoft.com • sales@scitechsoft.com • support@scitechsoft.com

SciTech Stereo DirectDraw Library

Version: 1.0
Revision: 0.12

Revision Date: Friday, June 30, 2000

This document defines the interface for a programming library that enables DirectDraw and Direct3D programmers to implement universal stereo support in their programs. This library supports a number of different formats for fullscreen stereo for LC shutter glasses, head mounted displays and stereo projectors. It also supports stereo in a window with LC shutter glasses. The library also supports just about every graphics adapter in existence, as well as just about every type of LC shutter glasses.

Intellectual Property

Copyright © 1999-2000 SciTech Software, Inc - All Rights Reserved. Duplication and distribution of this document is strictly prohibited without prior, written consent from SciTech Software, Inc.

While every precaution has been taken in the preparation of this document, SciTech Software assume no responsibility for errors or omissions, and make no warranties, expressed or implied, of functionality or suitability for any purpose.

Trademarks

All trademarks used in this document are property of their respective owners.

Acknowledgements

The following individuals or companies are acknowledged for contributions to the SciTech Stereo DirectDraw library and SDK: Page-Flip Emulation developed in cooperation with VRex Inc.; SDK examples provided by Dave Milici; Stereo Bitmap samples provided by Chasm Graphics; Stereo Animation samples provided by Richard Horne.

Table of Contents

Introduction	1
What is the SciTech Stereo DirectDraw Library?.....	1
Programming with Stereo DirectDraw.....	1
Standard Runtime Library Locations	1
Initializing Stereo DirectDraw.....	2
About Stereo DirectDraw formats	2
<i>Page flipped mode</i>	3
<i>Interlaced mode</i>	4
<i>Interleaved mode</i>	5
<i>Side-by-Side mode</i>	5
<i>Above-Below mode</i>	5
Scheduling the Stereo Buffer Flip.....	7
Refresh Rate Control.....	8
The Stereo DirectDraw Control Panel	8
Stereo DirectDraw Reference.....	11
Function Reference.....	11
<i>Reference Entry Template</i>	11
<i>DDS_exit</i>	Error! Bookmark not defined.
<i>DDS_getBlankIntervalLines</i>	Error! Bookmark not defined.
<i>DDS_getFlipStatus</i>	Error! Bookmark not defined.
<i>DDS_getStereoMode</i>	Error! Bookmark not defined.
<i>DDS_getWindowedStereoMode</i>	Error! Bookmark not defined.
<i>DDS_init</i>	Error! Bookmark not defined.
<i>DDS_restoreDisplayMode</i>	Error! Bookmark not defined.
<i>DDS_scheduleFlip</i>	Error! Bookmark not defined.
<i>DDS_scheduleFlip7</i>	Error! Bookmark not defined.
<i>DDS_start</i>	Error! Bookmark not defined.
<i>DDS_start7</i>	Error! Bookmark not defined.
<i>DDS_stereoOff</i>	Error! Bookmark not defined.
<i>DDS_stereoOn</i>	Error! Bookmark not defined.
<i>DDS_stop</i>	Error! Bookmark not defined.
<i>DDS_waitTillFlipped</i>	Error! Bookmark not defined.
Data Structure Reference	Error! Bookmark not defined.
<i>DDS_errorType</i>	Error! Bookmark not defined.
<i>GA_glassesTypeFlags</i>	Error! Bookmark not defined.
<i>GA_stereoModeType</i>	Error! Bookmark not defined.
<i>GA_winStereoModeType</i>	Error! Bookmark not defined.
Index.....	37

Introduction

This document contains the SciTech Stereo DirectDraw library reference.

What is the SciTech Stereo DirectDraw Library?

The SciTech Stereo DirectDraw library provides Windows DirectDraw-based applications with universal support for stereoscopic display devices.

Stereoscopic liquid crystal (LC) shutter glasses are a cheap, easy solution for getting real 3D stereoscopic imaging out of a standard PC with any standard monitor. LC shutter glasses work by constantly blanking out video information for each eye in a sequential fashion, allowing the user to see the left image for a fraction of a second followed by the right image, followed by the left image again etc. In order to make LC shutter glasses work effectively on PC based graphics controllers, some mechanism for changing the displayed video information at every vertical retrace is necessary. New hardware is available that will do this automatically, and the graphics device driver defines the software interface necessary to allow applications to use these new hardware features. The graphics driver will also emulate this field-sequential functionality for hardware lacking such capability.

This library supports LC shutter glasses as well as head-mounted displays (HMDs) and stereo projector systems implementing various stereo display formats through a common API.

Programming with Stereo DirectDraw

This chapter describes in detail the issues application software developers will face when developing code to use the SciTech Stereo DirectDraw libraries.

Standard Runtime Library Locations

SciTech Stereo DirectDraw relies upon the SciTech Nucleus Graphics Architecture device drivers to provide direct support for multiple graphics devices. The code necessary to load the SciTech Nucleus Graphics Architecture device drivers (graphics.bpd) is handled automatically by the SciTech Stereo DirectDraw interface library (ddstereo.lib).

Applications developed using SciTech Stereo DirectDraw are required to install the SciTech Nucleus graphics device driver file (graphics.bpd) in a subdirectory named 'drivers' below the application directory. Below the 'drivers' directory there must be a 'config' directory used to store configuration files. In the configuration directory you should ship the monitor database file (monitor.dbx) and the configuration options file (options.dat). In the application directory you should ship the Portability Manager helper VxD along with the DirectDraw Stereo control panel application. The following describes the runtime files provided and their locations with your application program directories (where x:\app is the full path to where your application is installed):

File	Directory Location
Nucleus device drivers	x:\app\drivers\graphics.bpd
Monitor database	x:\app\drivers\config\monitor.dbx
Configuration Options	x:\app\drivers\config\options.dat
Portability Manager VxD	x:\app\pmhelp.vxd
Stereo Control Panel	x:\app\ddscpl.exe

Note however that if the user has installed a copy of SciTech Display Doctor on their system, the SciTech Nucleus device drivers provided with SciTech Display Doctor will be used in preference to the ones in your application directory. Also note that the stereo control panel you ship will also bring up the standard SciTech Display Doctor stereo preferences page as opposed to the default standalone stereo control panel.

In the unlikely event that a conflict between the device drivers shipped with your application and SciTech Display Doctor arises, the user may also set the NUCLEUS_PATH environment variable to override the location of the device driver files for your application. This should be done on the command line, or using a batch file before your application is started (not in the AUTOEXEC.BAT file!). For

instance a small batch file such as the following can be used for 'compatibility' mode:

```
@echo off
set NUCLEUS_PATH=.
myapp.exe
```

Initializing Stereo DirectDraw

Initializing the SciTech Stereo DirectDraw library is performed by calling `DDS_init()` with the device index for the display controller to activate. In a typical application, the primary display controller with device index 0 is used.

As part of the initialization sequence, the SciTech Nucleus device driver must be able to be loaded from the 'drivers' directory relative to the application directory. If the SciTech Nucleus device driver cannot be loaded, `DDS_init()` will return an error code. Refer to the `DDS_errorType` enumeration for a list of possible error codes.

If the SciTech Nucleus device driver has been loaded properly, the SciTech Stereo DirectDraw library will complete initialization as long as there is a valid license to use the Nucleus driver. If the application expects to load an end-user licensed version of Nucleus, the call to `DDS_init()` is sufficient to complete validation of the license. If the application is using an ISV developer license for SciTech Nucleus, an ISV license file must be compiled into the application as a C data structure. This C data structure is then used to register SciTech Nucleus via a call to `GA_registerLicense()`, which must be made prior to initialization via `DDS_init()`. To obtain a software vendor license for SciTech Stereo DirectDraw, contact SciTech sales directly at <sales@scitechsoft.com>.

Once the Stereo DirectDraw library has been initialized, other Stereo DirectDraw APIs may be called. Note also that you should only even call `DDS_init()` once when your application is loaded, and `DDS_exit()` once when your application exits.

About Stereo DirectDraw formats

There are a number of different stereo modes available in the SciTech Stereo DirectDraw library to support the various types of stereo viewing devices. Liquid Crystal Shutter (LCS) glasses will select page-flipped or interlaced stereo modes, since the left- and right-eye display information will be generated on sequentially alternating video fields. Head-Mounted Displays (HMD) will select side-by-side or line-interleaved stereo modes, since the display information will be routed to separate left- and right-eye LCD viewing panels. Furthermore, the stereo modes are setup differently depending if the display is in fullscreen or windowed mode.

By default the selection of the type of stereo device attached, and the stereo mode used should always be done via the SciTech Stereo DirectDraw control panel application. Your application should determine what type of stereo mode is being

used, and if necessary make any internal adjustments to account for this (although internally the libraries are designed to take care of the details for you).

Page flipped mode

In Page-Flipped stereo mode, the left and right visual components are output as contiguous fields in a progressive-scan display mode. This may be used in either fullscreen or windowed display modes. In Page-Flipped stereo mode all buffers are created by the application with the full width and height of the primary surface.

Page-Flipped stereo mode is generally the preferred mode for fullscreen and windowed display modes, as you get the same detail in stereo as you would normally get in non-stereo modes. Page-Flipped stereo does however require high refresh rate non-interlaced display modes, lots of offscreen memory and high performance graphics hardware (if you are doing 3D stereo). Hence Page-Flipped stereo is usually a good choice for fullscreen modes, not so good in windowed modes if the user does not have a monitor capable of very high refresh rates (120Hz) at regular desktop resolutions.

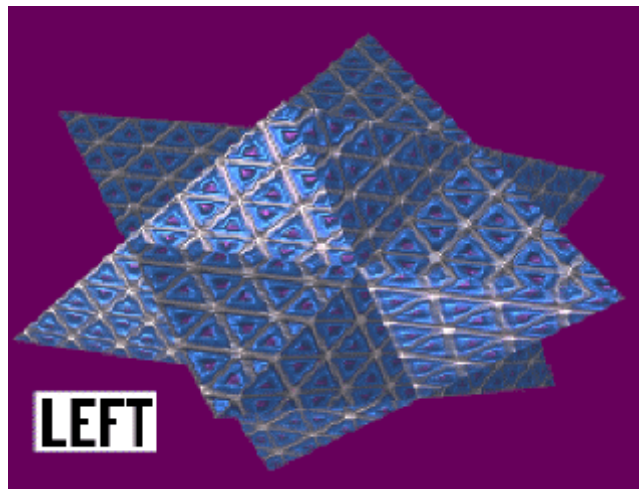


Figure 1a: Page-Flipped Stereo Field viewed through Left Eye Shutter

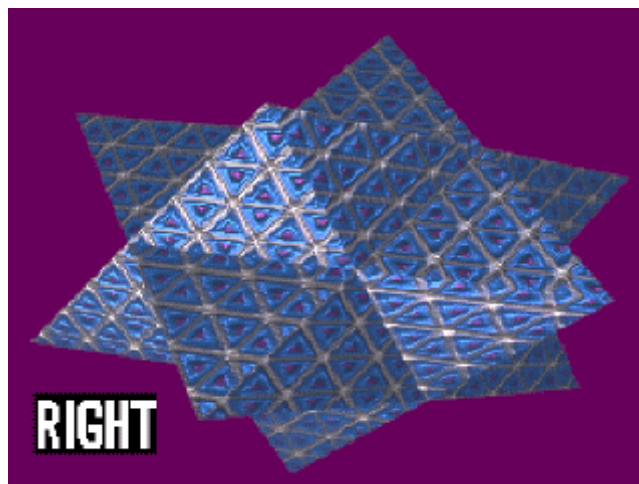


Figure 1b: Page-Flipped Stereo Field viewed through Right Eye Shutter

Interlaced mode

In Interlaced stereo mode, the left and right visual components are composited into an alternate-line format which coincide with the even- and odd-numbered scan line fields output in an interlaced display mode. This may be used in either fullscreen or windowed display modes. In this mode two stereo back buffers are created by the application with the full width and half the height of the display mode. All rendering to the stereo buffers should be done using double the aspect ratio of the primary surface. The stereo library will automatically take care of interleaving the stereo buffers onto the primary surface.

Interlaced stereo modes only use half the display memory and half the refresh rate of Page-Flipped stereo modes. Hence interlaced stereo mode are useful in situations where the user does not have enough graphics memory available for full page flipped stereo, or where the user's monitor is not capable of achieving high non-interlaced refresh rates at the desired resolution. Interlaced stereo modes usually provide reasonable results at desktop resolutions, but don't look very good at low game resolutions (such as 640x480).

Developer Note: If using Direct3D for rendering, be sure that the primary surface is not specified as a Direct3D rendering target, ie, DDSCAPS_3DDEVICE should be omitted from the surface description. Otherwise 3D accelerators which use tiled memory organization will produce images which are incompatible with interlaced display mode, and the resulting images will appear to be "mangled" by the display controller.

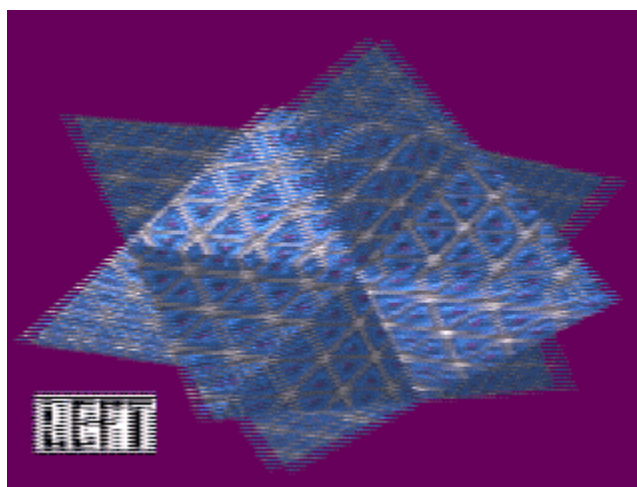


Figure 2: Interleaved/Interlaced Format Stereo Fields

Interleaved mode

In Interleaved stereo mode, the left and right visual components are composited into an alternate-line format the same way as interlaced stereo mode, except that the display is in a non-interlaced progressive-scan mode. This is to drive the separate LCD panels used by some types of head-mounted displays. This is typically used in fullscreen display mode for optimal viewing, but may also be used in windowed display mode if the Windows desktop is set at an appropriate resolution. In this mode two stereo back buffers are created by the application with the full width and half the height of the display mode. All rendering to the stereo buffers should be done using double the aspect ratio of the primary surface. The stereo library will automatically take care of interleaving the stereo buffers onto the primary surface.

Side-by-Side mode

In Side-by-Side stereo mode, the left and right visual components composited adjacent to each other in order to drive the separate LCD panels for particular types of head mounted displays. This is only used in fullscreen display modes. In this mode a full size buffer is created for the primary surface, which you may also optionally attach a flipping back buffer to (if you want no tearing). The stereo buffers are also created by the application but with half the width and the full height of the primary surface. All rendering to the stereo buffers should be done using half the aspect ratio of the primary surface. The stereo library will automatically take care of blitting the stereo back buffers to the primary surface and flipping the display if an attached back buffer is present.

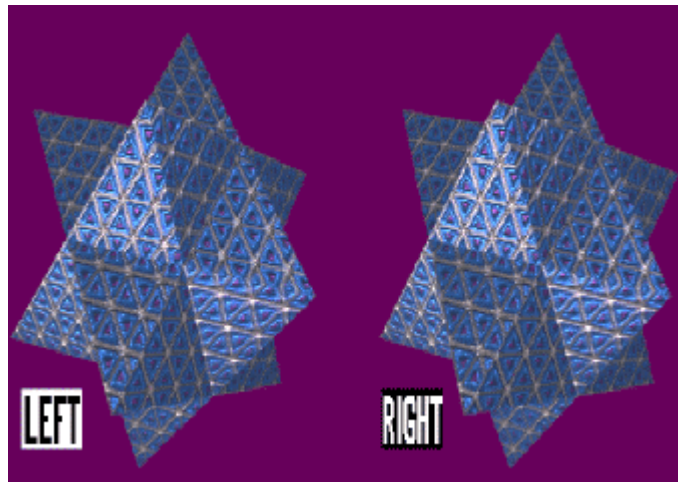


Figure 3: Side-by-Side Format Stereo Fields

Above-Below mode

In Above-Below stereo mode (sometimes called the “Over/Under” mode), the left and right visual components are composited into a fullscreen visual format one above the other. In this mode a full size buffer is created for the primary surface, which you may also optionally attach a flipping back buffer to (if you want no tearing). The stereo buffers are also created by the application but with the full width and half the height of the primary surface. All rendering to the stereo buffers

should be done using double the aspect ratio of the primary surface. The stereo library will automatically take care of blitting the stereo back buffers to the primary surface and flipping the display if an attached back buffer is present.

This mode is generally used for specific types of LCS glasses, such that the display output is routed through an external sync-doubling device. This device effectively produces sequentially alternating fields at half the original screen resolution. For example 1280x1024 mode at 60 Hz would be re-generated as (theoretically) 1280x512 at 120 Hz. Since the external sync-doubler device is inserting an extra vertical blanking interval into the display output signal, a black band is drawn inbetween the “above” and “below” visual components to coincide with the extra blanking interval. In the previous example, the actual sync-doubled output would be more like 1280x492 at 120 Hz when accounting for an extra blank interval height of 40 lines.

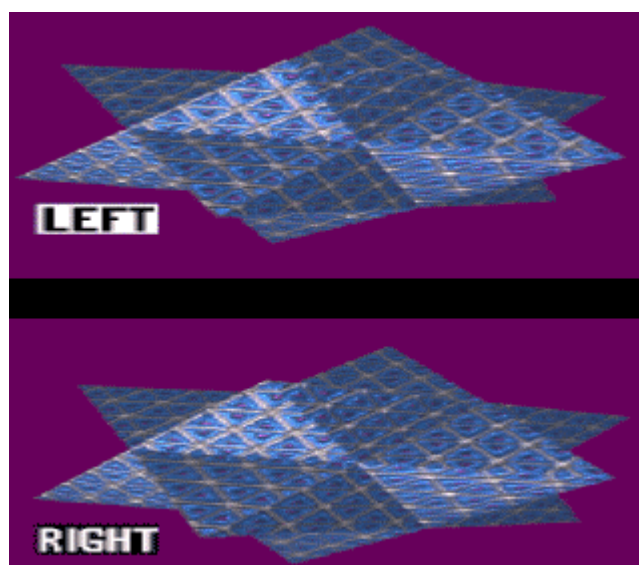


Figure 4: Above-Below Format Stereo Fields

Anaglyph mode

In Anaglyph stereo mode, the left and right visual components are filtered red and blue and then composited into a single image. For optimum viewing with red and blue passive eyewear, the left and right image components should be rendered in (or converted to) a monochromatic color space where all RGB channels contain the same average grayscale values. Otherwise color information will be lost, especially in the green channel, and the resulting appearance may be unsatisfactory. In this mode all stereo buffers are created with normal aspect ratio, and the stereo library will automatically perform the filtering and compositing.

Developer Note: The stereo library will only be able to support Anaglyph mode for chipsets which support AND and OR logical raster operations (ROPs). Otherwise the masking would have to be performed on a per-pixel basis, which would slow down rendering considerably. In such cases it would be better for the application to

perform such emulation, perhaps in conjunction with grayscale conversion, and only if speed were not an issue.

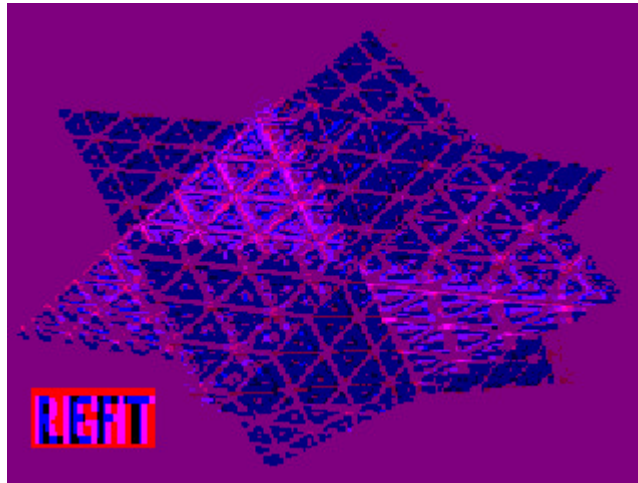


Figure 5: Anaglyph Format composite Stereo Fields

Scheduling the Stereo Buffer Flip

In page-flipped stereo modes, the display controller is automatically flipping between two pages in video memory at a constant rate. One page is used for the left eye image and the other for the right eye image. Hence the two flipping pages produce the alternating fields for viewing with LCS glasses. This means that at least two more pages are used for drawing while the visible stereo pages are flipping. A total of four pages are used for real-time rendering, so the stereo version of double-buffering is typically called quad-buffering.

When it becomes time to update visible stereo pages, the application should pass the new pair of left and right buffers to `DDS_scheduleFlip()`, and the Stereo DirectDraw driver will load these buffer start addresses into the display controller at the next left/right flip. Note that the new pages will not actually be updated until the next left-eye flip occurs, and then they will be updated as a pair. This keeps the stereo display pair in sync to avoid temporal parallax problems.

Note that if only four buffers are used in this manner, the application should wait until it is safe to draw into the previously display stereo buffer pair. The APIs `DDS_getFlipStatus()` and `DDS_waitTillFlipped()` support this scheme. High-performance applications which do not want to wait in this manner may implement a total a six buffers instead of four if sufficient video memory is available. This would then be the stereo equivalent of triple-buffered animation.

If your application is running in a Page-Flipped stereo mode, the same `DDS_scheduleFlip()` API is used, and you pass in the left and right buffers that you previously created at with the correct dimensions. Internally the libraries will take

care of all the differences between the different stereo modes inside DDS_scheduleFlip() for you.

Refresh Rate Control

For optimal viewing with stereoscopic devices, the refresh rate for the selected stereo mode may also need to be changed. A high refresh rate is desirable for LCS glasses so as to avoid noticeable flickering, like 120 Hz. By contrast, a low refresh rate is desirable for HMDs so as to avoid LCD panel lagging.

By default the refresh rate used should be selected by the user using the SciTech Stereo DirectDraw control panel.

Note also that an interlaced display mode must be setup within Nucleus differently from a non-interlaced progressive-scan display mode. However the value passed to either API function refers to the actual field rate. For example, 1024x768 interlaced mode at 86 Hz means each even or odd field is output as 1024x384 at 86 Hz.

The Stereo DirectDraw Control Panel

The Stereo DirectDraw control panel allows the user to set the various stereo display mode and refresh rate parameters discussed earlier. When the application starts up, the default settings that the user has selected using your stereo control panel will be used as the defaults for your application.

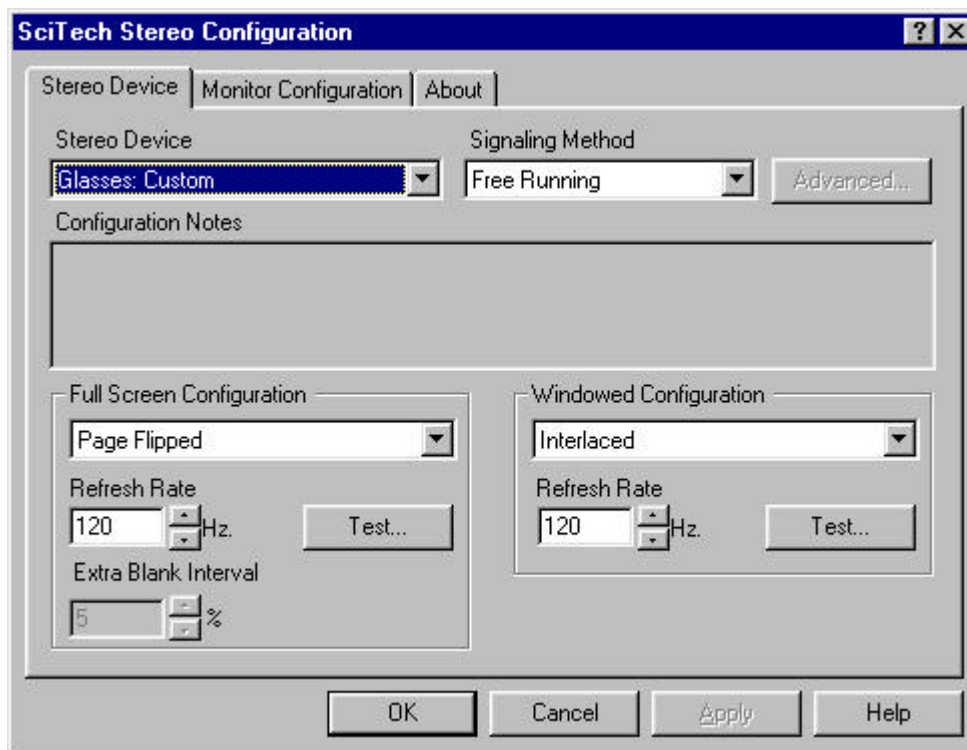


Figure 6: Stereo DirectDraw Control Panel.

Stereo Device

In Stereo Device selection, the user selects the type of stereo display device, whether it is LCS glasses, HMD helmet, stereo projector, or other stereoscopic device. Existing stereo devices are listed by vendor and/or brand name. Alternatively a custom device may be selected, such as for LCS glasses employing a previously unknown signalling scheme.

Signalling Method

In Signalling Method selection, the user selects the type of stereo sync signal used in conjunction with the selected LCS glasses or other field-sequential stereo device. These signalling methods might implement one of the following: I/O ports like the serial or parallel port connector; modulation of the vertical sync signal; drawing signal tags in the visible display area; or using the graphics board's stereo sync connector (if so equipped). Note that if you select a Stereo Device by brand name, the Signalling Method will automatically be selected to match. Some stereo devices might have multiple signalling methods available.

Advanced Options

In Advanced Options dialog, the user selects additional configuration data for programming the stereo sync signal, such as I/O port assignments, or V-Sync modulation. Note that if you select a Stereo Device by brand name, the Advanced Options will automatically be selected to match. However if you are using a different configuration or a custom stereo device, this is where you would program the specific signalling characteristics for on-vs-off and left-vs-right stereo sync data.

Full-Screen Configuration

In Full-Screen Configuration selection, the user selects the type of stereo display format to be effective in fullscreen modes, such as page-flipped, interlaced, interleaved, above-below, or side-by-side formats. Refresh Rate is effective only for full-screen stereo display modes. A Test button allows the user to preview the full-screen stereo display mode and refresh rate for up to 10 seconds, then will automatically restore the Windows desktop, which might be helpful if the selected mode or refresh rate was unintelligible.

Windowed Configuration

In Windowed Configuration selection, the user selects the type of stereo display format to be effective in windowed modes, such as page-flipped, interlaced, or interleaved formats. Refresh Rate is effective only for windowed stereo display modes, and may not necessarily be the same as the full-screen values. A Test button allows the user to preview the windowed stereo display mode and refresh rate for up to 10 seconds, then will automatically restore the Windows desktop in case the selected mode or refresh rate was unintelligible. The 10-second timeout may be cancelled by performing a move or resize on the test window, which might be

desirable for checking the interaction between stereo and non-stereo window regions.

Stereo DirectDraw Reference

This section contains the function reference and data structure reference for the SciTech Stereo DirectDraw library.

Function Reference

The section contains the function reference for the SciTech Stereo DirectDraw library.

Reference Entry Template

Summary of what the function does.

Syntax

```
<type> function( <type> parameter[,...] )
```

Prototype in

header.h

This lists the header file(s) containing the prototype for the function. The prototype of a function may be contained in more than one header file, in which case all the files would be listed, so use whichever one is more appropriate.

Parameters

Briefly describes each of the function parameters.

Return value

This section describes the value returned by the function (if any).

Description

This section describes what the function does, the parameters it takes and any details you might need to know in order to get full use out of the function.

See also

This section gives a list of other related functions that may be of interest.

DDS_exit

Closes down the DirectDraw Stereo API.

Declaration

```
void DDSAPI DDS_exit(void)
```

Prototype In

ddstereo.h

Description

This function closes down the DirectDraw Stereo API, and unloads any loaded device drivers. This function should be called when your application terminates.

See Also

DDS_init, DDS_stop

DDS_getBlankIntervalLines

Returns the extra vertical blanking interval used for above/below formats.

Declaration

```
int DDSAPI DDS_getBlankIntervalLines(void)
```

Prototype In

ddstereo.h

Return Value

Extra vertical blanking interval in scan lines.

Description

This function gets the height of the extra vertical blanking interval used for the Above/Below stereo format.

Note: *This function should be called **after** you have called `DDS_start` in order for the extra vertical blanking interval to be calculated for the current fullscreen display mode.*

See Also

DDS_scheduleFlip

DDS_getFlipStatus

Returns the status of the last scheduled stereo flip.

Declaration

```
int DDSAPI DDS_getFlipStatus(void)
```

Prototype In

ddstereo.h

Return Value

0 if flip has not occurred, non-0 if it has occurred.

Description

This function returns the status of the last scheduled software stereo flip. This bit is sticky and is set to 0 when the *DDS_scheduleFlip()* function is called, and reset to 1 when the flip actually occurs.

This function is applicable only for page-flipped stereo mode. In all other stereo modes, this function always returns 1.

See Also

DDS_stereoOn, *DDS_scheduleFlip*, *DDS_waitTillFlipped*, *DDS_stereoOff*

DDS_getStereoMode

Return the user selected fullscreen stereo mode.

Declaration

```
int DDSAPI DDS_getStereoMode(void)
```

Prototype In

ddstereo.h

Return Value

Mode selected by the user for fullscreen stereo modes.

Description

This function returns the mode selected by the user for fullscreen stereo modes. The mode itself is stored in the Nucleus GA_options structure (specific to each device in the system), and the values are determined by the GA_stereoMode enumeration. The known values at the time of writing are (consult the Nucleus reference documentation or header files for up to date information):

<i>gaStereoPageFlip</i>	Fullscreen page flipped stereo
<i>gaStereoAboveBelow</i>	Fullscreen above below format
<i>gaStereoSideBySide</i>	Fullscreen side by side format
<i>gaStereoInterleaved</i>	Fullscreen line interleaved stereo format
<i>gaStereoInterlaced</i>	Fullscreen interlaced stereo format

Note: You *must* first call *DDS_init* before you can use this function.

See Also

DDS_init, *DDS_getWindowedStereoMode*

DDS_getWindowedStereoMode

Return the user selected windowed stereo mode.

Declaration

```
int DDSAPI DDS_getWindowedStereoMode(void)
```

Prototype In

ddstereo.h

Return Value

Mode selected by the user for windowed stereo modes.

Description

This function returns the mode selected by the user for windowed stereo modes. The mode itself is stored in the Nucleus `GA_options` structure (specific to each device in the system), and the values are determined by the `GA_stereoMode` enumeration. The known values at the time of writing are (consult the Nucleus reference documentation or header files for up to date information):

<i>gaWinStereoInterlaced</i>	Windowed interlaced mode
<i>gaWinStereoInterleaved</i>	Windowed line-interleaved mode
<i>gaWinStereoPageFlip</i>	Windowed page-flipped mode

Note: You *must* first call `DDS_init` before you can use this function.

See Also

`DDS_init`, `DDS_getStereoMode`

DDS_init

Initialise the DirectDraw Stereo API for the specified display controller.

Declaration

```
int DDSAPI DDS_init(
    int deviceIndex)
```

Prototype In

ddstereo.h

Parameters

deviceIndex Index of display device to control (0 for primary)

Return Value

ddsOK on success, otherwise error code of type *DDS_errorType*.

Description

This function initialises the DirectDraw Stereo API, and loads the necessary device drivers. The main reason this function will fail is if there is an unsupported graphics adapter in the system. Check the return code to determine the failure condition.

Note: *This function should be called when your application initialises, and does not do anything related to the current display mode.*

During the life-cycle of the application, the DirectDraw Stereo APIs would be called in the following order:

```
DDS_init(deviceIndex);
DDS_start(hwndMain, lpDD, lpPrimarySurf);
DDS_stereoOn();
DDS_scheduleFlip(lpLeftSurf, lpRightSurf);
...
DDS_stereoOff();
DDS_stop();
DDS_restoreDisplayMode();
DDS_exit();
```

See Also

DDS_exit, DDS_start, DDS_getStereoMode, DDS_getWindowedStereoMode

DDS_restoreDisplayMode

Restores the original display mode active before stereo was activated

Declaration

```
void DDSAPI DDS_restoreDisplayMode(void)
```

Prototype In

ddstereo.h

Description

This function restores the active display mode that was present before the DirectDraw stereo mode was started. This function *must* be called by your application when shutting down your program, to properly ensure that the display mode is restore to it's previous state. This function should be called after you call *DDS_stop*, and before you call *DDS_exit*.

You should replace any code that calls IDirectDraw_RestoreDisplayMode in your code with code to call this function.

See Also

DDS_init, *DDS_stop*, *DDS_exit*

DDS_scheduleFlip

Schedule a new stereo display start address change.

Declaration

```
int DDSAPI DDS_scheduleFlip(
    DDS_LPDDDSURF lpLeftSurf,
    DDS_LPDDDSURF lpRightSurf)
```

Prototype In

ddstereo.h

Parameters

lpLeftSurf Left display surface to make active
lpRightSurf Right display surface to make active

Return Value

DirectDraw error code (either DD_OK or DDERR_SURFACELOST)

Description

This function schedules a new left and right surfaces to become active on the next vertical retrace for the LC shutter glasses. This is used for double and triple buffering when stereo page-flipped mode is active.

If the system is currently running in interleaved or interlaced stereo mode, this function performs an interleaved blit from the left and right surfaces to the primary surface to construct the composite interlaced stereo image. This function expects that the left and right DirectDraw surfaces have been rendered in a half-height aspect ratio, ie, doubled aspect ratio.

If the system is currently set in above/below or side-by-side stereo modes, this function performs an unscaled blit from the left and right surfaces to the primary surface to construct the composite stereo image. This function expects that the left and right DirectDraw surfaces have been rendered in an adjusted aspect ratio, half-height less extra vertical blanking interval for above/below format, or half-width for side-by-side format.

If the left and right surfaces are identical, and the selected stereo mode uses one of the half-height or half-width aspect ratio formats, then the left and right image components are expected to appear in corresponding viewport regions of the selected surface.

Note: *If DirectDraw reports that the surfaces are lost during a blit operation in this function, we return the DDERR_SURFACELOST error code. You should check this and do your regular lost surface handling if this function returns that error code.*

See Also

DDS_stereoOn, DDS_getFlipStatus, DDS_waitTillFlipped, DDS_stereoOff

DDS_scheduleFlip7

Schedule a new stereo display start address change.

Declaration

```
int DDSAPI DDS_scheduleFlip7(  
    DDS_LPDDSURF7 lpLeftSurf,  
    DDS_LPDDSURF7 lpRightSurf)
```

Prototype In

ddstereo.h

Parameters

LpLeftSurf Left display surface to make active
LpRightSurf Right display surface to make active

Return Value

DirectDraw error code (either DD_OK or DDERR_SURFACELOST)

Description

This function is equivalent to DDS_scheduleFlip() except using DirectDraw7 interfaces.

See Also

DDS_stereoOn, *DDS_getFlipStatus*, *DDS_waitTillFlipped*, *DDS_stereoOff*

DDS_start

Start the DirectDraw Stereo API for the current display mode.

Declaration

```
int DDSAPI DDS_start(
    DDS_HWND hwndMain,
    DDS_LPDD lpDD,
    DDS_LPDDSURF lpPrimarySurf)
```

Prototype In

ddstereo.h

Parameters

hwndMain	Main window handle (both fullscreen and windowed)
lpDD	Pointer to the application DirectDraw object
lpPrimarySurf	Pointer to the application primary DirectDraw surface

Return Value

ddsOK on success, otherwise error code.

Description

This function starts DirectDraw Stereo for the current display mode.

This function automatically determines if the application is running in fullscreen modes, or in a window on the desktop. If the application is running in a fullscreen mode, this function enables fullscreen stereo using the mechanism specified by the user on the Nucleus GA_options structure. By default when this function returns, the left and right buffers are set to both flip to the lpPrimarySurf surface (for page flip mode). You should call *DDS_scheduleFlip()* to actually start displaying stereo images.

If the application is running in a window, this function will start stereo using a high refresh rate interlaced or non-interlaced display mode. The stereo mode used is determined by the user selection in the Nucleus GA_options structure.

Note: *The refresh rate used will be as close as possible to the value specified in the GA_options structure stereoRefresh or stereoRefreshInterlaced parameter. The value is scaled back based on monitor and graphics card limitations. The user can also change this in the stereo control panel to a setting they prefer.*

Note: *This function must be called **after** you have changed display modes and created your primary DirectDraw surface.*

See Also

DDS_stop, DDS_exit

DDS_start7

Start the DirectDraw Stereo API for the current display mode.

Declaration

```
int DDSAPI DDS_start7(  
    DDS_HWND hwndMain,  
    DDS_LPDD7 lpDD,  
    DDS_LPDDSURF7 lpPrimarySurf)
```

Prototype In

ddstereo.h

Parameters

HwndMain Main window handle (both fullscreen and windowed)
LpDD Pointer to the application DirectDraw7 object
LpPrimarySurf Pointer to the application primary DirectDraw7 surface

Return Value

ddsOK on success, otherwise error code.

Description

This function is equivalent to `DDS_start()` except using DirectDraw7 interfaces.

See Also

DDS_start, *DDS_stop*, *DDS_exit*

DDS_stereoOff

Turns off the automatic software stereo flipping.

Declaration

```
void DDSAPI DDS_stereoOff(void)
```

Prototype In

ddstereo.h

Description

This function turns off the currently running stereo mode, putting the system back into 2D mode and turning off the LC shutter glasses.

If page-flipped stereo mode is active, software page-flipping is turned off, and the display is left showing the contents of the left-eye surface.

If interlaced stereo mode is active, interlaced display mode is turned off.

In all other stereo modes, this function has no effect.

See Also

DDS_stereoOn, DDS_scheduleFlip, DDS_getFlipStatus, DDS_waitTillFlipped

DDS_stereoOn

Enables stereo mode

Declaration

```
void DDSAPI DDS_stereoOn(void)
```

Prototype In

ddstereo.h

Description

This function enables stereo mode.

If page-flipped stereo mode is selected, this function enables stereo page-flipping operation to begin. By default when this function returns, the left and right buffers are set to both flip to the lpPrimarySurf surface. You should call *DDS_scheduleFlip()* to actually start displaying separate left and right stereo images.

If interlaced stereo mode is selected, this function will start stereo using a high refresh rate interlaced display mode.

In all other stereo modes, this function has no effect.

See Also

DDS_scheduleFlip, *DDS_getFlipStatus*, *DDS_waitTillFlipped*, *DDS_stereoOff*

DDS_stop

Stops stereo mode for the current display mode and cleans up.

Declaration

```
void DDSAPI DDS_stop(void)
```

Prototype In

ddstereo.h

Description

This function stops the DirectDraw Stereo mode, and cleans up ready for the next mode change. You should use this function to clean up after you switch back to GDI mode.

Note: *This function **must** be called before you clean up and destroy your DirectDraw objects.*

See Also

DDS_init, DDS_start, DDS_restoreDisplayMode

DDS_waitTillFlipped

Waits until the last scheduled flip operation has occurred.

Declaration

```
void DDSAPI DDS_waitTillFlipped(void)
```

Prototype In

ddstereo.h

Description

This function polls the status of the last scheduled flip operation, and waits until it has completed. However this function also does the important job of checking the RTC clock to ensure that it continues to run, and will re-start it if it has stopped for some reason. Hence you should call this function when you need to spin until the flip has occurred to catch problems when the RTC clock stops running.

This function is applicable only for page-flipped stereo mode. In all other stereo modes, this function returns immediately.

See Also

DDS_stereoOn, DDS_scheduleFlip, DDS_getFlipStatus, DDS_stereoOff

Data Structure Reference

This section contains the data structure and enumerations reference for library.

DDS_errorType

Declaration

```
typedef enum {  
    ddsOK,  
    ddsNotDetected,  
    ddsNotPOSTed,  
    ddsDriverNotFound,  
    ddsCorruptDriver,  
    ddsLoadMem,  
    ddsOldVersion,  
    ddsMemMapError,  
    ddsIOError,  
    ddsStereoDisabled,  
    ddsStereoStarted,  
    ddsUnableToFindCRTC,  
    ddsLockFailed,  
    ddsInternalError,  
    ddsUnknownMode,  
    ddsVideoMemError,  
    ddsDDrawCapsError,  
    ddsNotCertified,  
    ddsDDrawVersion,  
} DDS_errorType
```

Prototype In

ddstereo.h

Description

Error codes returned by *DDS_init* function to indicate driver load status if loading the device driver failed. Most of these error codes are just duplicates of the codes returned by the Nucleus device driver loader library.

Members

<i>ddsOK</i>	No error
<i>ddsNotDetected</i>	Hardware not detected
<i>ddsNotPOSTed</i>	Hardware has not been POSTed
<i>ddsDriverNotFound</i>	Driver file not found
<i>ddsCorruptDriver</i>	File loaded not a driver file
<i>ddsLoadMem</i>	Not enough memory to load driver
<i>ddsOldVersion</i>	Driver file is an older version
<i>ddsIOError</i>	General I/O error
<i>ddsMemMapError</i>	Could not map physical memory areas
<i>ddsStereoDisabled</i>	Stereo support is disabled
<i>ddsStereoStarted</i>	<i>DDS_start</i> has already been called
<i>ddsUnableToFindCRTC</i>	Unable to compute valid CRTC timings using GTF
<i>ddsLockFailed</i>	DirectDraw lock of a surface failed
<i>ddsInternalError</i>	Internal error in the DirectDraw stereo library
<i>ddsUnknownMode</i>	Unknown stereo mode
<i>ddsVideoMemError</i>	Not enough video memory to support stereo mode
<i>ddsDDrawCapsError</i>	DirectDraw capabilities unavailable for stereo mode
<i>ddsNotCertified</i>	Driver file is not certified for stereo
<i>ddsDDrawVersion</i>	DirectDraw version is not current

GA_glassesTypeFlags

Declaration

```
typedef enum {
    gaGlassesBlueCode,
    gaGlassesLPTPort,
    gaGlassesCOMPort,
    gaGlassesIOPort,
    gaGlassesVSync,
    gaGlassesH3D,
    gaGlassesLineBlanking,
    gaGlassesSyncDouble,
    gaGlassesFreeRunning,
    gaGlassesHardwareSync,
    gaGlassesH3DMod,
    gaGlassesH3DPlus,
    gaGlassesMask                = 0x0FFF,
    gaGlassesPortMask           = 0xF000,
    gaGlassesCOM1               = 0x1000,
    gaGlassesCOM2               = 0x2000,
    gaGlassesCOM3               = 0x3000,
    gaGlassesCOM4               = 0x4000,
    gaGlassesLPT1               = 0x5000,
    gaGlassesLPT2               = 0x6000,
    gaGlassesLPT3               = 0x7000,
} GA_glassesTypeFlags
```

Prototype In

nucleus/graphics.h

Description

This enumeration defines the types of known LC shutter glasses supported by Nucleus for software stereo support. The values for the COM or LPT port that the glasses are connected to, are only included when specific glasses types are selected that are known to be connected to a COM or LPT port. If the user selects a custom I/O port configuration with `gaGlassesIOPort`, it is expected the correct I/O port information is manually configured in the `GA_options` structure.

Members

<i>gaGlassesBlueCode</i>	LC glasses that use the blue code system
<i>gaGlassesLPTPort</i>	LC glasses toggled by parallel port
<i>gaGlassesCOMPort</i>	LC glasses toggled by serial port
<i>gaGlassesIOPort</i>	LC glasses toggled by custom I/O port
<i>gaGlassesVSync</i>	LC glasses toggled by vSync modification
<i>gaGlassesH3D</i>	LC glasses using original H3D key frames
<i>gaGlassesLineBlanking</i>	LC glasses using external line blanking
<i>gaGlassesSyncDouble</i>	LC glasses using external sync doubler
<i>gaGlassesFreeRunning</i>	LC glasses that are free running
<i>gaGlassesHardwareSync</i>	LC glasses connected to VESA stereo connector
<i>gaGlassesH3DMod</i>	LC glasses using modified H3D key frames
<i>gaGlassesH3DPlus</i>	LC glasses using H3D-Plus dongle
<i>gaGlassesMask</i>	Mask to find the glasses type
<i>gaGlassesPortMask</i>	Mask to find the port the glasses are attached to
<i>gaGlassesCOM1</i>	Indicates glasses are on serial port COM1
<i>gaGlassesCOM2</i>	Indicates glasses are on serial port COM2
<i>gaGlassesCOM3</i>	Indicates glasses are on serial port COM3
<i>gaGlassesCOM4</i>	Indicates glasses are on serial port COM4
<i>gaGlassesLPT1</i>	Indicates glasses are on parallel port LPT1
<i>gaGlassesLPT2</i>	Indicates glasses are on parallel port LPT2
<i>gaGlassesLPT3</i>	Indicates glasses are on parallel port LPT3

GA_stereoModeType

Declaration

```
typedef enum {
    gaStereoNone,
    gaStereoPageFlip,
    gaStereoAboveBelow,
    gaStereoSideBySide,
    gaStereoInterleaved,
    gaStereoInterlaced,
    gaStereoDualDisplay,
    gaStereoViewportFlip,
    gaStereoAnaglyph,
} GA_stereoModeType
```

Prototype In

nucleus/graphics.h

Description

Defines the type of fullscreen stereo mode supported by the end user system. The modes are described below:

The `gaStereoPageFlip` mode is the preferred mode for fullscreen display modes, and uses fullscreen page flipping with quad buffering. The system is put into a high refresh rate non-interlaced display mode. Four display pages are used, and each vertical retrace the screen flips between the two left and right buffers while the application draws to the hidden left and right buffers. In this mode all buffers are created by the application with the full width and height of the primary surface, and one of those buffers should be the primary surface. None of the buffers need to be attached in a `DirectDraw` flip ring.

The `gaStereoAboveBelow` mode is an alternate mode for fullscreen display modes, and uses one or two fullsize buffers for page flipping, and two half height back buffers for the stereo images. In this mode a full size buffer is created for the primary surface, which you may also optionally attach a flipping back buffer to (if you want no tearing). The stereo buffers are also created by the application but with the full width and half the height of the primary surface. All rendering to the stereo buffers should be done using double the aspect ratio of the primary surface. The stereo library will automatically take care of blitting the stereo back buffers to the primary surface and flipping the display if an attached back buffer is present.

The `gaStereoSideBySide` mode is an alternate mode for fullscreen display modes, and uses one or two fullsize buffers for page flipping, and two half width back buffers for the stereo images. In this mode a full size buffer is created for the primary surface, which you may also optionally attach a flipping back buffer to (if you want no tearing). The stereo buffers are also created by the application but with half the width and the full height of the primary surface. All rendering to the stereo buffers should be done using half the aspect ratio of the primary surface. The stereo library

will automatically take care of blitting the stereo back buffers to the primary surface and flipping the display if an attached back buffer is present.

The `gaStereoInterleaved` mode is a mode for fullscreen display modes. The graphics controller is put into a low refresh rate non-interlaced display mode, and the scanlines from the left and right buffers are interleaved together in horizontal scanlines. In this mode two stereo back buffers are created by the application with the full width and half the height of the display mode. All rendering to the stereo buffers should be done using double the aspect ratio of the primary surface. The stereo library will automatically take care of interleaving the stereo buffers to the primary surface.

The `gaStereoInterlaced` mode is a mode for fullscreen display modes. The graphics controller is put into a high refresh rate interlaced display mode, and the scanlines from the left and right buffers are interleaved together in horizontal scanlines. In this mode two stereo back buffers are created by the application with the full width and half the height of the display mode. All rendering to the stereo buffers should be done using double the aspect ratio of the primary surface. The stereo library will automatically take care of interleaving the stereo buffers to the primary surface.

The `gaStereoDualDisplay` mode is a mode that uses two display controllers. One controller displays the left eye image, and another controller displays the right eye image.

The `gaStereoViewportFlip` mode is a variation of the fullscreen page-flipped display mode where the left and right image components are expected to be rendered as viewports on the same surface. The graphics controller is put into a high refresh rate non-interlaced display mode at half the vertical resolution of the currently set display mode, and the upper and lower halves of the display surface are automatically page-flipped. In this mode a single stereo back buffer is created by the application with the full width and height of the primary surface, and the left and right image components are rendered into upper and lower viewports. All rendering to the stereo viewports should be done using double the aspect ratio of the primary surface.

The `gaStereoAnaglyph` mode is where the left and right image components are filtered with red and blue colors for viewing with anaglyph glasses.

Members

<i>gaStereoNone</i>	Stereo is not supported on this system
<i>gaStereoPageFlip</i>	Fullscreen page flipped stereo
<i>gaStereoAboveBelow</i>	Fullscreen above below format
<i>gaStereoSideBySide</i>	Fullscreen side by side format
<i>gaStereoInterleaved</i>	Fullscreen line interleaved stereo format
<i>gaStereoInterlaced</i>	Fullscreen interlaced stereo format
<i>gaStereoDualDisplay</i>	Fullscreen dual display stereo format
<i>gaStereoViewportFlip</i>	Fullscreen viewport flipped stereo
<i>gaStereoAnaglyph</i>	Fullscreen red/blue anaglyph stereo

GA_winStereoModeType

Declaration

```
typedef enum {
    gaWinStereoNone,
    gaWinStereoInterlaced,
    gaWinStereoInterleaved,
    gaWinStereoPageFlip,
    gaWinStereoPageFlipLoRes,
    gaWinStereoAnaglyph,
} GA_winStereoModeType
```

Prototype In

nucleus/graphics.h

Description

Defines the type of windowed stereo mode supported by the end user system. The modes are described below:

The `gaWinStereoInterlaced` mode is a mode for displaying stereo in a window on the desktop. The graphics controller is put into a high refresh rate interlaced display mode, and the scanlines from the left and right buffers are interleaved together in horizontal scanlines. In this mode two stereo back buffers are created by the application with the full width and half the height of the application window. All rendering to the stereo buffers should be done using double the aspect ratio of the primary surface. The stereo library will automatically take care of interleaving the stereo buffers onto the application window surface.

The `gaWinStereoInterleaved` mode is a mode for displaying stereo in a window on the desktop. The graphics controller is put into a user selectable refresh rate non-interlaced display mode. The mode is basically the same as `gaWinStereoInterlaced`, but interlaced mode is not used.

The `gaWinStereoPageFlip` mode is a mode for displaying stereo in a window on the desktop. The graphics controller is put into a high refresh rate non-interlaced display mode, and the stereo left and right buffers are automatically composited into the application window every retrace interval. In this mode four stereo back buffers are created by the application with the full width and height of the application window. All rendering to the stereo buffers should be done using the full aspect ratio of the primary surface. The stereo library automatically takes care of blitting the buffers to the application window every retrace.

The `gaWinStereoPageFlipLoRes` mode is a variation of the windowed page-flipped display mode which uses less video memory resources internally than the default `gaWinStereoPageFlip` mode. It could be used as a fallback contingency mode when video memory resources for the current desktop resolution are marginal. The trade-off for using less video memory is that dirty rectangle management for both stereo and non-stereo display regions will not be updated synchronously between the left and right eyes. This low-res stereo mode may appear acceptable for simple

windowed applications, but less so with greater number of windows displayed, especially overlapping and popup windows.

The gaWinStereoAnaglyph mode is where the left and right image components are filtered with red and blue colors for viewing with anaglyph glasses.

Members

<i>gaWinStereoNone</i>	Stereo is not supported on this system
<i>gaWinStereoInterlaced</i>	Windowed interlaced stereo mode
<i>gaWinStereoInterleaved</i>	Windowed interleaved stereo mode
<i>gaWinStereoPageFlip</i>	Windowed page flipped stereo mode
<i>gaWinStereoPageFlipLoRes</i>	Windowed page flipped low-res stereo mode
<i>gaStereoAnaglyph</i>	Windowed red/blue anaglyph stereo

D

DDS_errorType, 19, 28
DDS_exit, 8, 19, 20, 22
DDS_getBlankIntervalLines, 9, 15
DDS_getFlipStatus, 16, 21, 23, 24, 26
DDS_getStereoMode, 17, 18, 19
DDS_getWindowedStereoMode, 17, 18, 19
DDS_init, 8, 10, 11, 12, 13, 14, 17, 18, 19, 20, 25, 28
DDS_restoreDisplayMode, 19, 20, 25
DDS_scheduleFlip, 9, 16, 19, 21, 22, 23, 24, 26
DDS_start, 9, 15, 19, 22, 25, 29
DDS_stereoOff, 16, 19, 21, 23, 24, 26
DDS_stereoOn, 16, 19, 21, 23, 24, 26
DDS_stop, 8, 19, 20, 22, 25
DDS_waitTillFlipped, 16, 21, 23, 24, 26

G

GA_glassesTypeFlags, 10, 30
GA_stereoModeType, 13, 33
GA_winStereoModeType, 14, 35

GA_winStereoModeType