Porting Guide

3Dfx Interactive Glide 2.3

Document Revision 2.3

April 1997

3Dfx Interactive, Inc.
4435 Fortran Drive
San Jose, CA 95134
408-935-4400
www.3dfx.com

Friendly Reminder

The 3Dfx Interactive Software Developers Kit

This document is part of the 3Dfx Interactive Software Development Kit (SDK). The 3Dfx Interactive SDK documentation consists of:

- **DOCS\GPGUIDE.DOC - *Porting Guide: Glide 2.3***
- DOCS\ATBRELNO.DOC - *Release Notes: ATB 0.5*
- DOCS\D3DRELNO.DOC - *Release Notes: 3Dfx Direct 3D™ 2.08*
- DOCS\GLRELNO.DOC - *Release Notes: Glide 2.3*
- DOCS\INSTALL.DOC - *Installation Guide: 3Dfx Interactive Reference Boards and Related Software 2.6*

Why is 3Dfx Changing the Glide API?

We, at 3Dfx Interactive, feel it is imperative to provide a fast, consistent and bug-free native graphics API that is compatible with Voodoo Graphics™, Voodoo Rush™ and future products. Our goal is to provide interoperability throughout our product lines.

In order for Glide to be a well-rounded API, it must provide support for windows. Glide 2.3 extends support for 3D graphics within a window. Now, end-users and developers can run level editors, games and plug-ins on Voodoo Rush in a Direct3D™ or a Glide window with the same great performance that can be experienced on Voodoo Graphics!

In addition, we now provide support for MS-DOS® DLLs, eliminating the requirement to re-compile MS-DOS-based applications. These minor changes will allow developers to stay focused on creating great 3D games and ease worries about porting to updated 3Dfx hardware. Customers will be assured that when they buy a game with the 3Dfx logo on the CD, they know it will work on 3Dfx hardware. This will make our end users happy, providing a win-win situation for all.

To meet these goals, we had to make some minor API changes to Glide. Our experience with porting games to Glide 2.3 has been exceptional. Generally, it takes less than 30 minutes to make the code changes. If you need assistance with the porting, please ask us. You can send mail to devsupport@3dfx.com.

General information

3Dfx Interactive introduced the Voodoo Rush at COMDEX '96. Voodoo Rush is a two-chip chipset designed to provide a 2D/3D solution, in conjunction with compatible 2D accelerator chips; it offers the same features and performance as in Voodoo Graphics.

The primary purpose of this application note is to assist developers creating Glide applications that are compatible between Voodoo Graphics and Voodoo Rush. The fundamental architectural differences between the two chips that are visible to the Glide programmer are:

1. Voodoo Graphics is a pass-through device that operates with any existing 2D graphics card. This implies that when the output of Voodoo Graphics is visible, the output of the 2D graphics card is NOT (on a single monitor configuration). Voodoo Rush shares a frame buffer with a 2D graphics device (e.g., Alliance ProMotion AT3D, Trident ProVidia 9865-1, Macronix MX86251 or Media Realty Technologies MR510). This means that the 2D and 3D graphics can be displayed simultaneously on a desktop.

2. The Voodoo Graphics presents a uniform linear frame buffer space with a fixed stride (or pitch) of 1024 pixels, regardless of the screen resolution of the Voodoo Graphics. In contrast, the Voodoo Rush's frame buffer pitch depends on the associated 2D chip's desktop resolution. Also, LFB writes do not go through the pixel processing pipeline on Voodoo Rush.

Despite the differences, it is possible to create both MS-DOS 32 and Win32 based Glide programs that are binary compatible between these two platforms if the application follows certain design guidelines. The actual device dependence is isolated to a DLL extension for both MS-DOS and Windows®, and the application does not need to special case the Voodoo Graphics or Voodoo Rush.

The Glide library has changed slightly to accommodate Voodoo Rush and to ensure compatibility with future 3Dfx technology.

Changes in GLIDE Programming Style for 32-bit MS-DOS Applications

There are 3 code changes that need to be made to MS-DOS applications:

1. Initialization
2. Linear Frame Buffer Access
3. Pass Through Control

Initialization

All calls to grSstOpen() need to be replaced with calls to grSstWinOpen().

The arguments from grSstOpen() map into grSstWinOpen() as follows:

```
hwnd              - alway 0 in the DOS space
screen_resolution - same as grSstOpen()
refresh_rate      - same as grSstOpen()
color_format      - same as grSstOpen()
origin_location   - same as grSstOpen()
nColBuffers       - same as grSstOpen()
nAuxBuffers       - 1 for Z/Alpha Buffer, 0 for none
```

A typical call to grSstOpen()as follows:

```
if ( !grSstOpen( GR_RESOLUTION_640x480,
                 GR_REFRESH_60Hz,
                 GR_COLORFORMAT_ARGB,
                 GR_ORIGIN_UPPER_LEFT,
                 GR_SMOOTHING_ENABLE,
                 2
        ) ) {
        /* throw exception */
}

will now read

if ( !grSstWinOpen( 0,
                    GR_RESOLUTION_640x480,
                    GR_REFRESH_60Hz,
                    GR_COLORFORMAT_ARGB,
                    GR_ORIGIN_UPPER_LEFT,
                    2,
                    1
        ) ) {
        /* throw exception */
}
```

Linear Frame Buffer Writes

All direct frame buffer accesses must now be done between paired calls to grLfbLock() and grLfbUnlock(). Computation of pixel addresses must take into account a variable stride. This is a change from frame buffer address computation in Glide 2.1.1. To find the pixel at location (x,y) given a pointer to the base of the frame buffer "PTR," the address of (x,y) is computed as:

```
xyPtr = ((char*)PTR)+(y*strideInBytes)+(x*bytesPerPixel);
```

The stride of the locked frame buffer is returned in the GrLfbINfo_t structure passed to grLfbLock().

Typical Glide 2.1.1 linear frame buffer code might read:

```
void setPixel( int x, int y, short color ) {
    short *pixel;

    grLfbWriteMode(GR_LFBWRITEMODE_565);
    pixel = grLfbGetWritePtr(GR_BUFFER_BACKBUFFER);
    grLfbBypassMode( GR_LFBBYPASS_DISABLE );
    grLfbOrigin( GR_ORIGIN_UPPER_LEFT );
    grLfbBegin();
    pixel = pixel + ( y * 1024 ) + x;
    *pixel = color;
    grLfbEnd();
    return;
}


Under Glide 2.3 the same code would read:

void setPixel( int x, int y, short color ) {
    GrLfbInfo_t info;

    info.size = sizeof( info );
    if ( grLfbLock( GR_LFB_WRITE_ONLY,
                    GR_BUFFER_BACKBUFFER,
                    GR_LFBWRITEMODE_565,
                    GR_ORIGIN_UPPER_LEFT,
                    FXTRUE,
                    &info ) ) {
        short *pixel;

        pixel  =
(short*)(((char*)info.lfbPtr)+y*info.strideInBytes+x*2);
        *pixel = color;

        grLfbUnlock( GR_LFB_WRITE_ONLY,
                     GR_BUFFER_BACKBUFFER );
    } else {
        /* throw exception */
    }
}
```

Important!

You may not call ANY other Glide API after issuing a call to grLfbLock() until you call grLfbUnlock().

- grLfbWriteRegion() and grLfbReadRegion() are stand alone rectangle BLTing routines. They are not to be called within a grLfbLock()/grLfbUnlock() pair.

Pass Through

On the Voodoo Graphics architecture, 3D rendering takes place in a frame buffer entirely separate from the 2D graphics subsystem. In Glide 2.1, there was a routine called grSstPassthruMode() that was used to select between Voodoo Graphics 3D rendering and the 2D. Voodoo Rush shares the frame buffer with a host 2D, and in that environment, there is no practical analogue to the mode of operation applications are accustomed to with the pass through architecture.

Under the pass through architecture, an application could freely toggle between the 2D and the 3D displays and operate on each frame buffer completely independently. On Voodoo Rush, for example, the frame buffer is shared by the 2D and the 3D engine, and therefore, the application must take into account that 2D state changes now affect the state of the 3D engine. When an application wants to take advantage of the 2D frame buffer in order to, for example, display a low resolution compressed cutscene, it must disable the 3D hardware. The hardware is disabled through the grSstClose() API. In order to return to 3D rendering, a new call to grSstWinOpen() must be issued and the entire 3D state must be restored INCLUDING THE CONTENTS OF THE TEXTURE MEMORY. This is because the state of the hardware (in particular graphics memory) cannot be guaranteed between calls to grSstWinClose() and grSstWinOpen(). Applications that expect to only run on pass through architectures can still toggle the VGA pass through state through calls to grSstControl().

In Glide 2.1, when an application needed to switch to the VGA, it would do something like:

```
/*
** Doing 3D Rendering
*/
.

/*
** Show Movie
*/
.
grSstPassthruMode( GR_PASSTHRU_SHOW_VGA );
renderMyMovie();
grSstPassthruMode( GR_PASSTHRU_SHOW_SST1 );
.
/*
** Do More 3D Rendering
*/
```

.

In Glide 2.3 the preceding screen should be implemented as:

```
/*
** Doing 3D Rendering
*/
.
/*
** Show Movie
*/
grSstWinClose();
int386( blah blah blah ); /* switch to Mode 13h for example */
renderMyMovie();
if ( !grSstWinOpen( 0,
                    GR_RESOLUTION_640x480,
                    GR_REFRESH_60Hz,
                    GR_COLORFORMAT_ARGB,
                    GR_ORIGIN_UPPER_LEFT,
                    2,
                    1
        ) ) {
   /* throw exception */
}
downloadWorkingTextureSet();   /* restore texture ram */
.
/*
** Do More 3D Rendering
*/
```

Changes in GLIDE Programming Style for Win32 Applications

Glide 2.3 (and all previous versions of Glide) is mutually exclusive with Microsoft Direct3D; Glide applications can not run simultaneously with D3D applications. If a running D3D application is detected when Glide is initializing, the Glide initialization will fail. Glide will also currently disallow running multiple simultaneous Glide applications.

Glide 2.3 now includes support for windowed rendering on the Voodoo Rush family. Previous versions of Win32 Glide were designed around the pass through architecture which was unable to render into a window and shared no display resources with the underlying operating system. With the introduction of Voodoo Rush, Glide applications may now render directly to the primary display surface. In addition to the changes to linear frame buffer access discussed in the MS-DOS section above, there are two critical sections of your code that have to change to work in the windows environment.

- Initialization
- Window event handling

Initialization

A Win32 Glide application may run in one of two primary modes. These are differentiated in your code by the argument you pass in the screen_resolution argument to grSstWinOpen().

Fullscreen

Note: This is the only mode available to Voodoo Graphics. Pass a valid Glide resolution in the screen_resolution argument to grSstWinOpen().

For example:

```
HWND hwnd = GetActiveWindow();

grSstWinOpen( hwnd,
              GR_RESOLUTION_640x480,
              GR_REFRESH_60Hz,
              GR_COLORFORMAT_ABGR,
              GR_ORIGIN_UPPER_LEFT,
              2, 1 );
```

Windowed

Pass an argument of GR_RESOLUTION_NONE in the screen_resolution argument to grSstWinOpen(). Glide will render to the client surface of the window pointed to by hwnd. This call will fail, if Glide is unable to allocate sufficient frame buffer RAM for the requested buffers. It is the responsibility of the application to manage the clip rectangle for the back buffer. Clipping to the window's primary surface is handled by Glide. You may not render to the front buffer in Windowed mode.

```
HWND hwnd = GetActiveWindow();

grSstWinOpen( hwnd,
              GR_RESOLUTION_NONE,
              GR_REFRESH_60Hz,
              GR_COLORFORMAT_ABGR,
              GR_ORIGIN_UPPER_LEFT,
              2, 1 );
```

Window Event Handling

Glide needs to be made aware of certain window events. This is accomplished by calling the grSstControl() API from within the message handling loop of your application window. The messages which you must capture are:

```
Window Message          grSstcontrol() Argument
==============          =======================
WM_SIZE                 GR_CONTROL_RESIZE
WM_MOVE                 GR_CONTROL_MOVE
WM_ACTIVATE             GR_CONTROL_ACTIVATE/GR_CONTROL_DEACTIVATE

long FAR PASCAL MainWndproc( HWND hWnd,
                             UINT message,
                             WPARAM wParam,
                             LPARAM lParam )  {
    switch( message )
    {
    case WM_MOVE:
        if (!grSstControl(GR_CONTROL_MOVE)) {
            PostMessage( hWndMain, WM_CLOSE, 0, 0 );
            return 0;
        }
        break;
    case WM_SIZE:
        if (!grSstControl(GR_CONTROL_RESIZE)) {
            PostMessage( hWndMain, WM_CLOSE, 0, 0 );
            return 0;
        }
        break;
    case WM_ACTIVATE:
        {
           WORD fActive = LOWORD(wParam);
           BOOL fMinimized = (BOOL) HIWORD(wParam);

           if ( ( fActive == WA_INACTIVE ) || fMinimized ) {
              grSstControl( GR_CONTROL_DEACTIVATE );
           } else {
               grSstControl( GR_CONTROL_ACTIVATE );
           }
       }
       break;
    return DefWindowProc(hWnd, message, wParam, lParam);

  } /* MainWndproc */
```

grSstStatus()

Although grSstStatus() still returns the contents of the status register on Voodoo Rush, the bitfield definitions have changed. Specifically, the Voodoo Rush status register no longer contains FIFO freespace. Applications which poll this register on Voodoo rush to determine FIFO status will fail.

Caveats

Glide 2.3 may execute in exactly one thread in a system at a time. Glide is not reentrant. You must take care not to call any Glide API from a Windows message loop except grSstControl().

Summary of API Changes

Summary of the API changes for Glide 2.3.

API	CHANGE	JUSTIFICATION
grLfbBegin	Removed	Old lfb paradigm has been abandoned
grLfbEnd	Removed	Old lfb paradigm has been abandoned
grLfbLock	Added	Changes the paradigm for LFB access, lfb access needed a notion of variable stride as well as a lockin notion for windowing systems.
GrLfbOrigin	Removed	Subsumed by lock
grLfbReadRegion	Added	You may not read-lock fb ram on an SLI system, we can, however read rectangles of fbram from an SLI system at the driver level.
GrLfbUnlock	Added	See grLfbLock
grLfbwriteMode	Removed	Subsumed by lock
grLfbWriteRegion	Added	Locks are to be discouraged whenever possible in favor of optimized host memory blts which can be hardware accelerated
grSstControl	Added	Used to inform GLIDE of system events
grSstOpen	Removed	Obsolete
grSstPassThru	Removed	The notion of passthru is out-moded in future hardware, this is subsumed into a more generalized grSstControl
grSstWinClose	Added	Allows Glide applications to shut down and restart 3D engine safely.
grSstWinOpen	Added	Allows for Glide In a Window
guFbReadRegion	Removed	Replaced by grLfbReadRegion
guFbWriteRegion	Removed	Replaced by grLFbWriteRegion

3DFX INTERACTIVE, INC. LICENSE AND CONFIDENTIALITY AGREEMENT

LICENSE AND CONFIDENTIALITY: 3Dfx Interactive, Inc. ("3Dfx") grants you the right to install the enclosed software and related documentation (collectively, the "Materials") onto a single computer for

your personal use. You may not use, copy, modify, sell, transfer or disclose any part of the Materials except as provided in this Agreement. You may only use the Materials in connection with the development of game titles, software products or demo software for 3Dfx products.

RESTRICTIONS

You may not:

Sublicense or permit simultaneous use of the Materials by more than one user;

Reverse engineer, decompile, or disassemble the enclosed software;

Use the Materials for any purpose other than developing game titles, software products, or demo software for any platform or products other than 3Dfx products. Without limiting the generality of the foregoing, you may not use or disclose all or any part of the Materials in connection with the development of products competitive with 3Dfx chips, drivers, APIs and other products;

Make copies of the Materials other than for back-up purposes, and you may not use the back-up copies other than as a replacement for the original copy. You must include on the back-up copies all copyright and other notices included on the Materials; and Export the Materials in violation of the export control laws of the United States of America and other countries.

GENERATED CODE: 3Dfx hereby grants to you the right to include the object code "runtime" version of the enclosed software in your software product for the 3Dfx platform (the "Generated Code") and the right to replicate and distribute (and have others replicate and distribute) such object code "runtime" version of the enclosed software worldwide, but only as part of the Generated Code. You agree to indemnify 3Dfx and its affiliates against any loss, liability or expense (including reasonable legal fees) arising out of or in connection with the use, marketing, licensing or sale of the Generated Code or the maintenance, support or other services or activities related thereto. TERMINATION: Upon any violation of any of the provisions of this Agreement, your right to use the Materials shall automatically terminate and you shall be obligated to return to 3Dfx or destroy all of your copies of the Materials. OWNERSHIP AND COPYRIGHT OF MATERIALS: Except for the license expressly granted hereunder, 3Dfx retains all rights, title and interests in and to the Materials and all copies thereof. The Materials are copyrighted and are protected by United States copyright laws and international treaty provisions. You acknowledge that the Materials are valuable trade secrets of 3Dfx. You may not remove the copyright and other proprietary rights notices from the Materials. You agree that this Agreement shall be retained with all printed and electronic copies the software and documentation constituting the Materials. You agree to prevent any unauthorized copying of the Materials. Except as expressly provided herein, 3Dfx does not grant any express or implied right to you under 3Dfx patents, copyrights, trademarks, or trade secret information.

NO WARRANTY; NO LIABILITY FOR DAMAGES: THE MATERIALS ARE PROVIDED "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF SATISFACTORY QUALITY, MERCHANTABILITY, NONINFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT SHALL 3DFX BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DIRECT OR INDIRECT DAMAGES, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, LOSS OF INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE MATERIALS, EVEN IF 3DFX HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Without limiting the generality of the foregoing, no warranty is made that the enclosed software will generate computer programs with the characteristics or specifications desired by you or that the Generated Code will be error-free.

THESE DISCLAIMERS OF WARRANTY CONSTITUTE AN ESSENTIAL PART OF THIS AGREEMENT. Because some jurisdictions prohibit the exclusion or limitation of liability for damages, the above limitation may not apply to you and you may have other legal rights that vary by jurisdiction.

NO SUPPORT: 3Dfx does not provide any support for the Materials. However, 3Dfx does have a newsgroup at its web site in which the enclosed software is discussed. 3Dfx does not, however, make any representations or warranties as to the accuracy of any statements or advice provided by any participant in the newsgroup.

US GOVERNMENT RESTRICTED RIGHTS. Use, duplication or disclosure by the U.S. government is subject to restrictions set forth in subparagraphs (a) through (d) of the Commercial Computer-Restricted Rights clause at FAR 52.227-19 when applicable, or in subparagraph ©(l)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, or at 252.211-7015, or to 3Dfx's standard commercial license, as applicable, and in similar clauses in the NASA FAR Supplement. Contractor/manufacturer is 3Dfx Interactive, Inc., 4435 Fortran Drive, San Jose, California 95134. OTHER AGREEMENT: If you have executed another license agreement with 3Dfx with respect to the Materials, then notwithstanding any other term in this Agreement the terms of that license agreement shall control your use of the Materials. MISCELLANEOUS. Subject to the immediately preceding paragraph, this Agreement represents the complete agreement concerning this license and may amended only by a writing executed by both parties. If any provision of this Agreement is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This Agreement shall be governed by California, U.S.A. law (except for conflict of law provisions). The application the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. "3Dfx Interactive", the 3Dfx Interactive logo, and other 3Dfx Interactive product names are trademarks, and in some jurisdictions may be registered trademarks, of 3Dfx Interactive or its affiliated companies. Other trademarks are the property of their respective owners. Copyright 1996, 1997 by 3Dfx Interactive, Inc. All rights reserved. 12/96 SDK

NOTICE REGARDING SOFTWARE

Any software that is made available to download from this site ("Software") is the copyrighted work of 3Dfx Interactive, Inc. and/or its suppliers. Use of the Software is governed by the terms of the end user license agreement, if any, which accompanies or is included with the Software ("License Agreement"). You will be unable to install any Software that is accompanied by or includes a License Agreement without first agreeing to the License Agreement terms. Reproduction or redistribution of the Software, including to any other server or location, not in accordance with the License Agreement is expressly prohibited.