Fastgraph® for Windows®

Version 6.02 Release Notes

Ted Gruber Software, Inc.
PO Box 13408
Las Vegas, NV 89112

(702) 735-1980 voice
(702) 735-4603 FAX

support@fastgraph.com
http://www.fastgraph.com

Copyright © 1995-2001 Ted Gruber Software, Inc.
All Rights Reserved.

Introduction

The Fastgraph 6.02 for Windows (FGW) maintenance update provides improved printer support, DirectX enhancements, floating point or integer texture coordinates, and other new functions. It also adds support for Delphi 6.0 and fixes all problems reported since the 6.01 release. The 6.02 update includes updated versions of the Fastgraph example programs that have changed in this release.

New versions of the *Fastgraph 6.0 User's Guide*, the *Fastgraph 6.0 Reference Manual*, and the Fastgraph 6.0 help file are not included here, but are available from http://www.fastgraph.com/help.html.

This distribution contains patch files, not full libraries or units. After you apply the patches, your FGW 6.01 libraries will be converted to version 6.02. Complete instructions for applying the update patches are provided later in this document.

The FGW 6.02 update will work only if you have installed FGW from the version 6.01 CD, or if you have installed FGW from the version 6.00 CD and applied the version 6.01 update.

In addition, the update will work properly only with the original libraries or unit files. If you have modified these files in *any way*, you must re-install the original libraries, update the original libraries, and then make your own modifications to the resulting libraries.

Before applying the FGW 6.02 update, you should make sure the disk drive on which the library or unit files reside has at least 500K bytes of free space (this space is needed only during the update process). After the update, all successfully patched files will be dated 07-18-01 (July 18, 2001) and have a 6:02 a.m. time stamp.

The files in this distribution are:

FGW602.PDF	This file
FGW602.RTP	Fastgraph 6.02 update patches
PATCH.EXE	RTPatch application utility program
FGWIN.H	Fastgraph 6.02 header file for C/C++
ExBuilder1.zip	Updated example programs for C++Builder 1
ExBuilder3.zip	Updated example programs for C++Builder 3/4/5
ExC.zip	Updated example programs for C/C++
ExDelphi.zip	Updated example programs for Delphi
ExMFC.zip	Updated example programs for MFC
ExPB.zip	Updated example programs for PowerBASIC
ExVB.zip	Updated example programs for Visual Basic

Fastgraph/Fonts and Fastgraph/Image Updates

Separate updates are available on the Fastgraph web page for Fastgraph/Fonts 6.02 and Fastgraph/Image 6.02. If you have either or both of these Fastgraph add-on products, you must also apply the patch updates for those products after you apply the FGW 6.02 update.

New Features Added in Fastgraph 6.02

Fastgraph 6.02 offers improved printer support, DirectX enhancements, floating point or integer texture coordinates, and other new capabilities. These features are described in the following sections, and also in the function summary presented later in this document.

Improved Printer Support

The new **fg_printdc()** function establishes the printer device context, which directs subsequent printing to the associated printer. Its only parameter is a handle to the printer device context; this is often obtained

through the Print dialog box. If you do not call **fg_printdc()**, or if you pass it a zero device context, printing will be directed to the default Windows printer. The Prdemo example has been updated to use **fg_printdc()**.

DirectX Enhancements

The new **fg_ddmemory()** function lets us specify if DirectDraw surfaces created with **fg_vballoc()** will reside in video memory or in system memory (by default, such surfaces are created in system memory). Video memory surfaces generally offer additional hardware acceleration. When both the source and destination surfaces are in video memory, DirectDraw can usually take advantage of hardware-accelerated blitting when copying between these surfaces. Passing 0 to **fg_ddmemory()** means DirectDraw surfaces subsequently created with **fg_vballoc()** will reside in system memory; passing any other value means they will reside in video memory. Note that **fg_ddmemory()** only applies to DirectDraw surfaces created with **fg_vballoc()**. The DirectDraw primary surface (and its associated back buffer, if present) will always reside in video memory.

The new **fg_ddrestore()** function provides an easier way to restore the DirectDraw primary surface and all other DirectDraw surfaces that reside in video memory. Such surfaces must be restored upon return from a task switch (Alt+Tab, for example). This function restores the surface memory, but *not* the surface contents.

The **fg_ddmemory()** and **fg_ddrestore()** functions are available only in Fastgraph's DirectX libraries.

Texture Coordinates

Fastgraph's texture mapping functions now support either integer or floating point texture coordinates. By default, Fastgraph's texture mapping functions assume their (u,v) texture coordinate arrays contain integer values. But when 3D clipping is applied to textures, floating point (u,v) coordinates can provide more accurate results. The new **fg_tmunits()** function has a single integer parameter that specifies if the (u,v) coordinate arrays use integer or single precision floating point values. If this parameter is zero, the texture mapping functions assume integer (u,v) coordinates; if it is any other value, they assume floating point (u,v) coordinates. Note specifically the use of single precision (32-bit) floating point texture coordinates here, not double precision (64-bit). Single precision floating point means the type `float` in C and C++, and `Single` in Delphi and Basic.

Other New Functions

The new **fg_setcolorrgb()** function establishes the current color by combining the effect of **fg_setcolor()** and **fg_maprgb()**. When used with a 256-color virtual buffer, **fg_setcolorrgb()** will use the closest matching color in the logical palette.

The new **fg_vbtcopy()** function copies a rectangular region from one virtual buffer to another, or to a non-overlapping region in the same virtual buffer, excluding any pixels whose value matches the specified transparent color. The transparent color is a logical palette index for 256-color virtual buffers, or an **fg_maprgb()** encoded color value for direct color virtual buffers. As with Fastgraph's other block transfer routines, no clipping is performed.

Applying the Fastgraph 6.02 Patch

Follow these steps to apply the Fastgraph 6.02 patch:

Step 1: Copy the files PATCH.EXE and FGW602.RTP to the directory where you've installed the Fastgraph 6.01 libraries or unit files, and make that directory your current directory. If you've installed FGW for more than one compiler or platform and the library files reside in different directories, you'll need to apply the patch from each such directory (see Step 5).

Step 2: If you're using Delphi, rename the DCU files as follows:

For Delphi 2.0:

```
RENAME FGWIN*.DCU *.D20
```

For Delphi 3.0:
```
RENAME FGWIN*.DCU *.D30
```
For Delphi 4.0:
```
RENAME FGWIN*.DCU *.D40
```
For Delphi 5.0:
```
RENAME FGWIN*.DCU *.D50
```
For Delphi 6.0:
```
RENAME FGWIN*.DCU *.D60
```

Delphi 6.0 unit files do not exist for Fastgraph 6.01, so the 6.02 update uses the Delphi 5.0 unit files for version 6.01 to create Delphi 6.0 unit files for version 6.02. Renaming the Fastgraph 6.01 Delphi 5.0 unit files to a D60 extension will create Fastgraph 6.02 Delphi 6.0 unit files. Renaming them to a D50 extension will create Fastgraph 6.02 Delphi 5.0 unit files.

Step 3: If you're using Watcom C/C++ 11, rename the library files as follows:

```
RENAME FGWWC32.LIB  FGWVC32.LIB
RENAME FGWWC32D.LIB FGWVC32D.LIB
```

Step 4: Apply the patch by entering

```
PATCH FGW602
```

from the DOS command line. The PATCH utility will update all Fastgraph 6.01 libraries and unit files found in the current directory.

Descriptive messages will appear as the individual patches are applied. When the PATCH command completes, it will display a summary showing how many files were updated and how many were "missing". The missing files do not indicate a problem but merely mean you haven't installed FGW support for that particular compiler or platform.

Step 5: If you've installed FGW for more than one compiler, you must repeat Steps 1 to 4 from each directory where the library or unit files are stored for a given compiler. For example, suppose you've installed FGW for Borland C++ (with library files in C:\BC5\LIB) and Delphi 5.0 (with unit files in C:\DELPHI5\LIB). First copy the files PATCH.EXE and FGW602.RTP to the C:\BC5\LIB directory and issue the PATCH command to update your Borland C++ libraries. Following this, update the Delphi units by copying PATCH.EXE and FGW602.RTP to C:\DELPHI5\LIB and issue the PATCH command again. The order in which you apply patches for different compilers does not matter.

Step 6: If you're using Delphi, rename the unit files back to their original names:

For Delphi 2.0:
```
RENAME FGWIN*.D20 *.DCU
```
For Delphi 3.0:
```
RENAME FGWIN*.D30 *.DCU
```
For Delphi 4.0:
```
RENAME FGWIN*.D40 *.DCU
```
For Delphi 5.0:
```
RENAME FGWIN*.D50 *.DCU
```
For Delphi 6.0:
```
RENAME FGWIN*.D60 *.DCU
```

Step 7: If you're using Watcom C/C++ 11, rename the library files back to their original names:

```
RENAME FGWVC32.LIB  FGWWC32.LIB
RENAME FGWVC32D.LIB FGWWC32D.LIB
```

Step 8: If you're using Visual Basic, you must now update the FGWin.bas and FGWinD.bas module files. Copy the files PATCH.EXE and FGW602.RTP to the directory where you've installed the module files, and make that directory your current directory. Then apply the patch again as done in Step 4.

Step 9: If you're using PowerBASIC, you must now update the FGWin.inc and FGWinD.inc include files. Copy the files PATCH.EXE and FGW602.RTP to the directory where you've installed the include files, and make that directory your current directory. Then apply the patch again as done in Step 4.

Step 10: If you're using C/C++ or C++Builder, copy the FGWIN.H header file to a directory where the compiler normally searches for such files. The FGWIN.H file supplied in this distribution replaces the same file from earlier versions of Fastgraph for Windows.

Step 11: After applying the patch, you may delete all extra copies of the PATCH.EXE and FGW602.RTP files. You should keep one copy of these files in case you later install libraries for other compilers from the Fastgraph 6.01 CD.

New Versions of the Fastgraph 6.0 Examples

We recommend updating your FGW example programs with the versions supplied in this distribution. You can do this as follows:

C/C++	unzip ExC.zip into \FGW6\Examples\C
C++Builder 1	unzip ExBuilder1.zip into \FGW6\Examples\Builder1
C++Builder 3/4/5	unzip ExBuilder3.zip into \FGW6\Examples\Builder3
MFC	unzip ExMFC.zip into \FGW6\Examples\MFC (use -d switch)
Delphi	unzip ExDelphi.zip into \FGW6\Examples\Delphi
PowerBASIC	unzip ExPB.zip into \FGW6\Examples\PB
Visual Basic	unzip ExVB.zip into \FGW6\Examples\VB

Note that the above zip files contain only the example programs that have changed with this release.

If you've made any custom changes to the Fastgraph examples, you may first want to rename your modified examples or move them elsewhere.

Problems Corrected in Fastgraph 6.02

For Delphi, the maximum number of virtual buffers has been increased to 256 (as it is for other compilers).

Support for DirectX versions 2 and 3 did not work. This problem was introduced in Fastgraph 6.01.

Direct3D z-buffer creation now works for DirectX implementations that require z-buffers to use the same bit depth as the primary surface, even if they offer higher z-buffer bit depths.

All arc, circle, and ellipse functions can now draw larger objects. Previously, these functions could not draw objects if the horizontal radius multiplied by the vertical radius was greater than 1024*1024.

The native Gouraud shading functions did not always draw large polygons correctly when using 24-bit and 32-bit virtual buffers.

The **fg_aviplay()** and **fg_showavi()** functions now work with 16-color, high color, and 32-bit true color uncompressed AVI files.

Passing zero for the **fg_avimake()** compressor parameter did not create an uncompressed AVI file.

The **fg_aviplay()** function had a palette problem that occurred when creating a 16-color BMP file with **fg_makebmp()** after displaying a 16-color AVI file.

Calling **fg_kbtest(0)** did not report the key state correctly on some systems.

The **fg_tmfree()** function did not fully release the texture handle when using Fastgraph's DirectX library with Fastgraph's 3D rendering.

The **fg_tmtransparency()** declaration was not correct in the Visual Basic module files and the PowerBASIC include files.

The **fg_vballoc()** function did not work after calling **fg_tmdefine()** when using Fastgraph's DirectX library with Direct3D.

Passing a negative value to **fg_vbfree()** resulted in a memory leak.

The DirectX version of **fg_vbtzcopy()** could not attach a color key with certain DirectX implementations. In such cases, **fg_vbtzcopy()** will now use Fastgraph's own blitting.