QuickDraw GX and the Macintosh Environment

This chapter describes those aspects of QuickDraw GX that relate specifically to the Macintosh Toolbox, Macintosh programming environment, and Macintosh image data format. The chapter addresses the following topics:

- the Macintosh interface to QuickDraw GX
- the QuickDraw–to–QuickDraw GX translator

Before reading this chapter, you should be generally familiar with QuickDraw GX and QuickDraw GX objects, as described in the chapter "Introduction to QuickDraw GX" in *Inside Macintosh: QuickDraw GX Objects*. Additional specific information related to view ports and view devices is in the "View-Related Objects" chapter in *Inside Macintosh: QuickDraw GX Objects*.

Because this chapter describes the interface between QuickDraw GX and the rest of the Macintosh Toolbox, it uses many terms defined elsewhere. For a general picture of the Macintosh Toolbox, see *Inside Macintosh: Overview* or the introductory chapter of *Inside Macintosh: Macintosh Toolbox Essentials*. For information on Macintosh windows, see the chapter "Window Manager" in *Inside Macintosh: Macintosh Toolbox Essentials*. Mouse location and mouse handling is described in the chapter "Event Manager" in *Inside Macintosh: Macintosh Toolbox Essentials*. QuickDraw, QuickDraw coordinates, the QuickDraw picture format, picture comments, graphics ports, and Macintosh graphics devices are all described in *Inside Macintosh: Imaging With QuickDraw*.

About QuickDraw GX and the Macintosh Environment

QuickDraw GX provides a number of useful functions that assist your application development on the Macintosh computer. The Macintosh interface provides functions specific to the Macintosh platform that allow you to use information provided by other parts of Macintosh system software. The QuickDraw–to–QuickDraw GX translator allows you to convert from QuickDraw pictures to QuickDraw GX objects.

The Macintosh Interface

Most QuickDraw GX functions are designed for implementation on any platform. However, there are specific functions that are wrappers for Macintosh system software functions or have meaning only in the Macintosh environment. QuickDraw GX contains **Macintosh interface functions** to convert between QuickDraw and QuickDraw GX coordinate systems, find the mouse position in QuickDraw GX coordinates, associate view ports with Macintosh windows, map between view devices and Macintosh `GDevice` records, and intercept drawing commands to a view port.

The QuickDraw–to–QuickDraw GX Translator

The **QuickDraw–to–QuickDraw GX translator** can be used to convert QuickDraw drawing commands into QuickDraw GX objects. It allows you to create a set of QuickDraw GX **objects** that have a similar appearance to that intended by the original QuickDraw picture. This capability is useful, for example, for importing QuickDraw data from the Clipboard into a QuickDraw GX–based application.

It is important to note that the translator does not provide a completely faithful, pixel-by-pixel mapping of the image defined by the QuickDraw commands. However, it does closely approximate the original image, and you can even control the closeness of the approximation. In most cases the differences are subtle and not apparent to the eye.

Using QuickDraw GX in the Macintosh Environment

This section describes how you can

- determine QuickDraw GX versions
- use the Macintosh interface functions
- use the QuickDraw–to–QuickDraw GX translator

Testing for the Presence and Version of QuickDraw GX

You can use the `Gestalt` function in your application to determine which parts of QuickDraw GX are installed and their version numbers. The `Gestalt` function returns a 32-bit value that indicates the version of QuickDraw GX that is installed. The graphics and typography part of QuickDraw GX has one version number, the printing part of QuickDraw GX has another, and there is an overall version number that applies to all of QuickDraw GX. In addition, you can determine if the debugging version or the non-debugging version of QuickDraw GX is installed, and if the installed version is native to PowerPC system software.

To determine the current version of QuickDraw GX in general, you call the `Gestalt` function with the `gestaltGXVersion` selector. The function returns a value indicating the version of QuickDraw GX printing currently installed. For version 1.0, the value returned is 0x00010000.

To determine the current version of the graphics and typography parts of QuickDraw GX, you call the `Gestalt` function with the `gestaltGraphicsVersion` selector. The function returns a value indicating the version of QuickDraw GX graphics and typography currently installed. For version 1.0, the value returned is 0x00010000.

To determine the current version of the printing part of QuickDraw GX, you call the `Gestalt` function with the `gestaltPrintingMgrVersion` selector. The function returns a value indicating the version of QuickDraw GX printing currently installed. For version 1.0, the value returned is 0x00010000.

To determine if the debugging or non-debugging version of QuickDraw GX is currently installed, or if the installed version is native to PowerPC system software, you call the `Gestalt` function with the `gestaltGraphicsAttr` selector. The `gestaltGraphicsisDebugging` attribute value is returned if the debugging version of QuickDraw GX is installed. The `gestaltGraphicsisLoaded` attribute value is returned if the non-debugging version of QuickDraw GX is installed. The `gestaltGraphicsIsPowerPC` attribute value is returned if the installed version of QuickDraw GX is PowerPC-native. The return value can be any combination of those attributes.

Listing 1-1 uses the `gestaltGraphicsVersion` and `gestaltPrintingMgrVersion` selectors to determine whether QuickDraw GX graphics and typography as well as QuickDraw GX printing are installed. This listing also uses the `gestaltGraphicsAttr` selector to determine whether the installed version of QuickDraw GX is the debugging or non-debugging version.

Listing 1-1 Determining the presence and features of QuickDraw GX

```
Boolean QuickDrawGXAvailable(Boolean *pIsDebugging)
{
   Boolean returnValue = false;
   long theFeature, flags;
   if(Gestalt(gestaltGraphicsVersion, &theFeature) == noErr)
      {
         returnValue = true;
         if (Gestalt(gestaltPrintingMgrVersion, &theFeature) ==
                                                         noErr)
            gQDGXPrintingInstalled = true;
      }
   else
      returnValue = false;
   if (Gestalt(gestaltGraphicsAttr, &theFeature) == noErr)
      {
         if (flags & gestaltGraphicsisDebugging)
            pIsDebugging = true;
         else
            pIsDebugging = false;
      }
   return returnValue;
}
```

For additional details concerning the use of the `Gestalt` function to determine features of the QuickDraw GX environment, see the chapter “Gestalt Manager” in *Inside Macintosh: Operating System Utilities.*

Using the Macintosh Interface Functions

The QuickDraw GX Macintosh interface functions allow you to integrate QuickDraw GX with the Macintosh Toolbox. These functions allow you to

- create and use view ports associated with Macintosh windows
- retrieve a QuickDraw graphics device associated with a QuickDraw GX view device.
- convert between QuickDraw and QuickDraw GX coordinate systems
- intercept QuickDraw GX drawing functions for a view port

Creating and Using View Ports with Macintosh Windows

QuickDraw GX drawing takes place in **view ports.** You can associate a view port with a window in order to clip drawing to the window's visible region. Once you've created a window, you can create a view port that is associated with that window by the use of the `GXNewWindowViewPort` function. When you attach a view port to a window, you guarantee that all the shapes that you draw to the view port will be drawn in the correct location within the window, even when the window is moved. You also guarantee that if the window is underneath others that the QuickDraw GX drawing will be clipped to the QuickDraw GX window's visible region. You can attach a view port to your window with the call

```
windowParentViewPort = GXNewWindowViewPort(theWindow);
```

The resulting `windowParentViewPort` contains the view port attached to the window. You cannot change either the mapping or the **clip** of the window view port. If you need to do either—for example, if you need to control position within the view port (as when scrolling) or clip drawing within the window (so you don't draw over scroll bars), you need to create a **child view port** of your window view port and draw only to it. Child view ports, **view port hierarchies,** and how to use them are described in detail in the chapter "View-Related Objects" in *Inside Macintosh: QuickDraw GX Objects*.

Once you've created a view port, you can determine the view port that is associated with a specific window by using the `GXGetWindowViewPort` function. If you haven't associated a view port to that window, the function returns `nil`.

You can find out which window is associated with a view port by using the `GXGetViewPortWindow` function. The function returns `nil` if the view port is not associated with any window.

The `GXNewWindowViewPort` function is described on page 1-24. The `GXGetWindowViewPort` function is described on page 1-26.
The `GXGetViewPortWindow` function is described on page 1-25.

Using View Devices With Graphics Devices

On the Macintosh, every monitor gets a **graphics device,** described by a `GDevice` record. So when QuickDraw GX creates a screen **view device** for each monitor, there is already a graphics device for it. The `GXGetViewDeviceGDevice` and `GXGetGDeviceViewDevice` functions link the two worlds together, so that you can work with either description of a display device.

These functions work only with Macintosh system graphics devices. If you have a screen view device, you can call the `GXGetViewDeviceGDevice` function to get the graphics device that corresponds to that view device. If you create your own offscreen view device, it will not have an associated graphics device. Likewise, there is no view device associated with an offscreen `GDevice` record.

The `GXGetViewDeviceGDevice` function is described on page 1-27; the `GXGetGDeviceViewDevice` function is described on page 1-28.

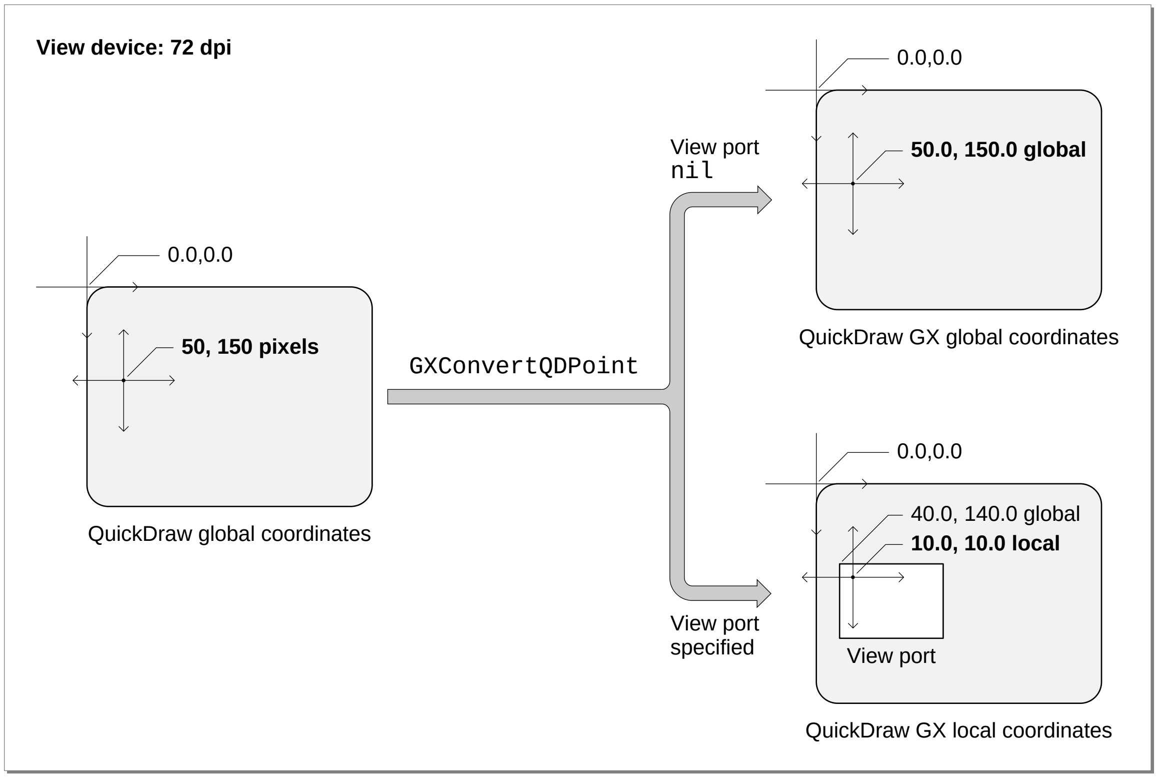
Converting From QuickDraw to QuickDraw GX Coordinates

QuickDraw GX provides several functions that involve conversion of locations on the QuickDraw coordinate plane into locations expressed in QuickDraw GX **local** or **global coordinates.**

Converting from QuickDraw Global to QuickDraw GX Local or Global Coordinates

You can use the `GXConvertQDPoint` function to convert a point having QuickDraw global coordinates to either QuickDraw GX global or QuickDraw GX local coordinates. If a view port is specified in the function's parameters, the QuickDraw point coordinates are converted to the corresponding QuickDraw GX local coordinates. If the view port parameter is `nil`, the QuickDraw point coordinates are converted to corresponding QuickDraw GX global coordinates. Figure 1-1 shows how the `GXConvertQDPoint` function converts a point having QuickDraw global coordinates of (50, 150) pixels on a monitor to QuickDraw GX coordinates in points (in which 1 point equals 1/72 inch).

Figure 1-1 Converting from QuickDraw global to QuickDraw GX local and global coordinates



When the view port parameter is `nil`, the QuickDraw global coordinates are converted to QuickDraw GX global coordinates (50.0, 150.0). When the view port is specified, the QuickDraw global coordinates are converted to QuickDraw local coordinates (10.0, 10.0) when the view port is located at QuickDraw GX global coordinates (40.0, 140.0). The local coordinates are local relative to the specified view port.

The `GXConvertQDPoint` function is described on page 1-29. For additional information about QuickDraw GX local, global, and device space, see the chapter "View-Related Objects" in *Inside Macintosh: QuickDraw GX Objects.*

Obtaining Mouse Location in Global Coordinates

The `GXGetGlobalMouse` function returns the location of the Macintosh cursor (mouse) in QuickDraw GX global coordinates. If a QuickDraw GX view device has a resolution of 72 dpi and a cursor is located at point (500, 150) pixels in QuickDraw coordinates, the `GXGetGlobalMouse` function would return the QuickDraw GX coordinates (500.0, 150.0) in points. If the resolution of the QuickDraw GX view device is 144 dpi and the cursor were at (1000, 300) pixels, the `GXGetGlobalMouse` function would again return coordinates (500.0, 150.0). No matter what the resolution of the device, the QuickDraw GX global coordinates are the same for a cursor located at a given absolute position.

The `GXGetGlobalMouse` function is described on page 1-30. For additional information about local, global, and device spaces, see the chapter "View-Related Objects" in *Inside Macintosh: QuickDraw GX Objects*.

Obtaining Mouse Location in Local Coordinates

For a given view port, you can use the `GXGetViewPortMouse` function to obtain the mouse position in the coordinate system (local coordinates) of that view port. This function takes any scaling of local space into account; if, for example, you have a zoomed-in view, the coordinates would be relative to the zoomed coordinate system.

If you obtain the mouse point in QuickDraw global coordinates, you can take the result of the `GXGetViewPortMouse` function and immediately turn it into a shape. You can use the `GXNewShape` function with the returned point as the shape origin, and QuickDraw GX will draw the shape at the point where the mouse is located with the correct scale. If the scale factor is 10, the shape is drawn enlarged by a factor of 10.

The `GXGetViewPortMouse` function is described on page 1-30. For additional information about local, global, and device spaces, see the chapter "View-Related Objects" in *Inside Macintosh: QuickDraw GX Objects*.

Intercepting Drawing Calls to a View Port

The `GXSetViewPortFilter` function causes QuickDraw GX to intercept all drawing function calls to a specified view port and pass them instead to an application-defined callback function that you supply. You can use the filter function to perform actions other than screen drawing, or perhaps to collect information about them.

QuickDraw GX uses this function to install a view port filter for printing. When a page is open and a call is made to draw a shape, instead of actually drawing it to the screen, the printing view port filter records (spools) it to the print file. You can use this kind of function if you want to achieve a similar result or if you otherwise want to manipulate shapes that would be drawn to a view port.

When you use the `GXGetViewPortFilter` function, you get back what you set with the `GXSetViewPortFilter` function. If you want to get rid of your view port filter, use the `GXSetViewPortFilter` function and specify a `nil` filter function.

The `GXSetViewPortFilter` function is described on page 1-31. The `GXGetViewPortFilter` function is described on page 1-32. The application-defined callback filter function is described on page 1-40.

Using the QuickDraw–to–QuickDraw GX Translator

The QuickDraw–to–QuickDraw GX translator converts QuickDraw drawing commands into QuickDraw GX shapes. There are two ways to use the translator:

- The first way is to pass the translator a handle to a QuickDraw picture. The translator returns a QuickDraw GX picture shape that approximates the original QuickDraw picture. The section "Using the Translator With QuickDraw Pictures" beginning on page 1-20 describes how to use the translator in this way.
- The second way is to use a pair of functions to install and remove the translator. After you install the translator in a given graphics port, it intercepts all subsequent QuickDraw drawing commands sent to that port and converts them to QuickDraw GX shapes. After you are finished converting, you remove the translator. The section "Installing and Removing the Translator" beginning on page 1-21 describes how to use the translator in this way.

The next section, "Factors in Translation," describes how translation works and how you can influence it by setting various translation parameters.

IMPORTANT
In order to use the QuickDraw-to-QuickDraw GX translator, you first must have called the `GXInitPrinting` function. The `GXInitPrinting` function is described in the core printing features chapter of *Inside Macintosh: QuickDraw GX Printing.* ▲

Factors in Translation

This section describes some of the factors that influence the translation process, and how you can manipulate them.

Graphics Port and View Port

The translation from QuickDraw to QuickDraw GX takes into account the current QuickDraw grafPort origin. Therefore, each resulting QuickDraw GX shape incorporates, either in its shape geometry or in its transform mapping, the origin of the graphics port that was active at the time of translation.

The QuickDraw GX shape that results from the translation must be associated with a view port. This can be accomplished by

- setting the view port for each shape
- setting the view port for the parent picture shape of the individual shapes contained in a picture

Scaling During Translation

The translator allows you to scale the QuickDraw data as it is converted. For example, you can use scaling to convert from a screen resolution of 72 dpi to a printer resolution of 300 dpi. You specify the scaling factor in the form of source and destination rectangles.

Also, in order to allow the translator to properly scale dash picture comments and other items, you can supply a pair of integer scale factors, which may be different in the x and y directions. The scale factors for both the source and destination rectangles and the pattern-stretch parameters are usually the destination resolution divided by the screen resolution (72 dpi), rounded to the nearest integer. Typical examples are shown in Table 1-1.

Table 1-1 Translation scaling factors

Source	Destination	Scale Factor
72×72	72×72	1×1
72×72	72×80	1×1
72×72	144×144	2×2
72×72	150×150	2×2
72×72	300×300	4×4

Translation Options

When you translate QuickDraw data to QuickDraw GX shapes, you specify one or more **translation options.** You can use either the default translation option provided by QuickDraw GX or a combination of the other available options. Some translation options provide simpler and faster translations, but with a resulting loss of pixel-for-pixel matching. Table 1-2 lists and describes the available translation options; the constants are defined in the `gxTranslationOptions` enumeration.

Table 1-2 Translation options settings

Constant	Value	Explanation
`gxDefaultOptionsTranslation`	0x0000	This is the default setting used for translation. This option generates the most accurate representation of the QuickDraw data that the translator is capable of producing.
`gxOptimizedTranslation`	0x0001	This option allows for optimizations to be applied during translation. For example, a sequence of QuickDraw lines can be combined into one polygon. In most cases, this results in the generation of a smaller number of QuickDraw GX shapes.
`gxReplaceLineWidthTranslation`	0x0002	The width of a resulting QuickDraw GX line is the average of the original pen's width and height. This option also affects the way in which the SetLineWidth `PicComment` is interpreted. The LaserWriter driver scales the current line width with the newly specified picture comment. The translator normally uses the LaserWriter mechanism. When you specify this option, the translator uses a mechanism in which the line is replaced with the newly specified width; this mimics the behavior of the LaserWriter SC driver.
`gxSimpleScalingTranslation`	0x0004	This option causes the translator to scale data from source resolution to destination resolution by using a simple multiplication, which is incorporated into the shape's transform. The translator makes no attempt to compensate for this increase in resolution. The resulting scaled image will not render the original QuickDraw data accurately, but will be similar to what QuickDraw would have produced when it attempted to scale the data.
`gxSimpleGeometryTranslation`	0x0008	This option results in a translation of QuickDraw data without taking into account the QuickDraw hanging pen. Normally the translator reproduces a QuickDraw triangle, for example, as a 6-sided or 7-sided polygon. This option sacrifices accuracy in order to produce an image that draws faster with QuickDraw GX and can be more useful for pen-based output devices. For example, QuickDraw lines become QuickDraw GX lines with flat endcaps. This option also turns on the simple lines translation and the simple scaling translation.

Table 1-2 Translation options settings (continued)

Constant	Value	Explanation
`gxSimpleLinesTranslation`	0x000C	This option results in simple geometry and scaling. The translator maintains the width of lines that are at an angle. Because QuickDraw uses a hanging pen, a diagonal line appears to be thicker than a horizontal or vertical line with the same pen size. Using this option causes the line width to be the same as the pen width, at the expense of the accuracy of the original QuickDraw data. This option also turns on the simple scaling translation and the simple geometry translation.
`gxLayoutTextTranslation`	0x0010	Normally the translator turns off layout-shape-specific capabilities when translating QuickDraw text into layout shapes. This option restores layout features such as default glyph substitutions. This results in a more attractive text; however it can be different from the original QuickDraw data.
`gxRasterTargetTranslation`	0x0020	This option causes PostScript picture comments to be discarded. The bitmap proxies sent along with such comments are preserved.
`gxPostScriptTargetTranslation`	0x0040	This option causes PostScript picture comments to be incorporated as tags attached to picture shapes. The bitmap proxies sent along with such comments are discarded.
`gxVectorTargetTranslation`	0x0080	This option causes PostScript picture comments to be discarded. The bitmap proxies sent along with such comments are preserved. Also, when this option is combined with the option `gxOptimizedTranslation`, lines are preserved and not combined in thick framed polygons.

How Option Settings Affect Translation of Lines

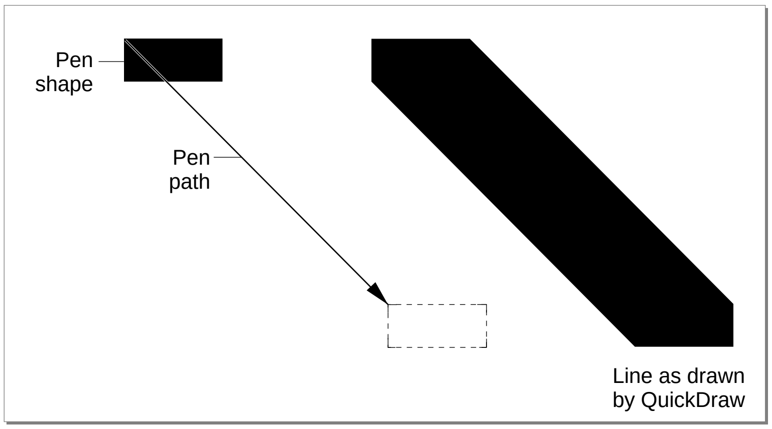
The translation of QuickDraw lines is affected by the translation options setting you choose. Consider the simple line generated by the QuickDraw commands given in Listing 1-2.

Listing 1-2 QuickDraw commands to draw a simple line

```
PenSize(5, 3);
MoveTo(100, 40);
LineTo(120, 70);
```

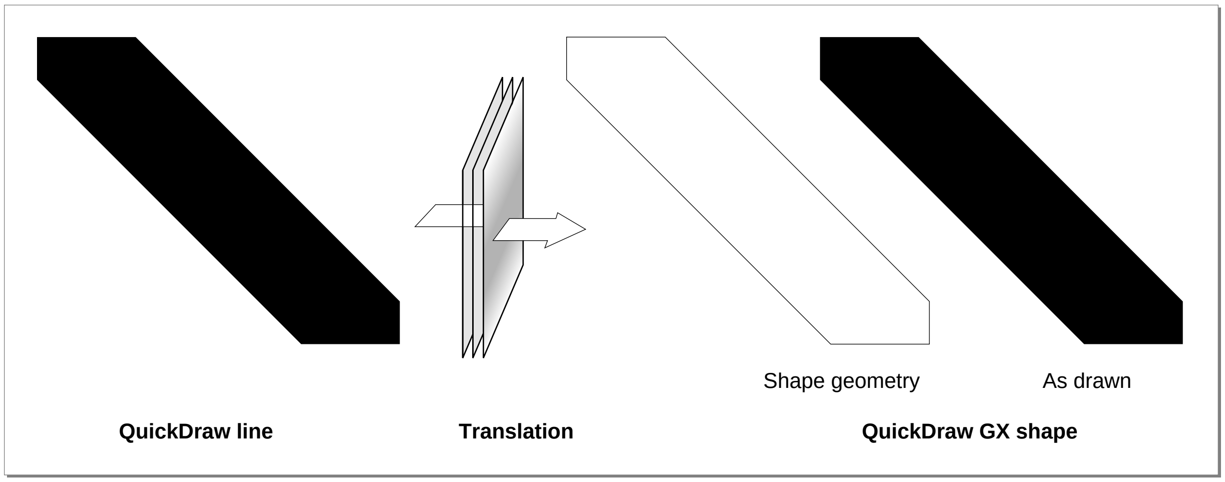
The QuickDraw commands in Listing 1-2 produce the line shown in Figure 1-2.

Figure 1-2 A QuickDraw line



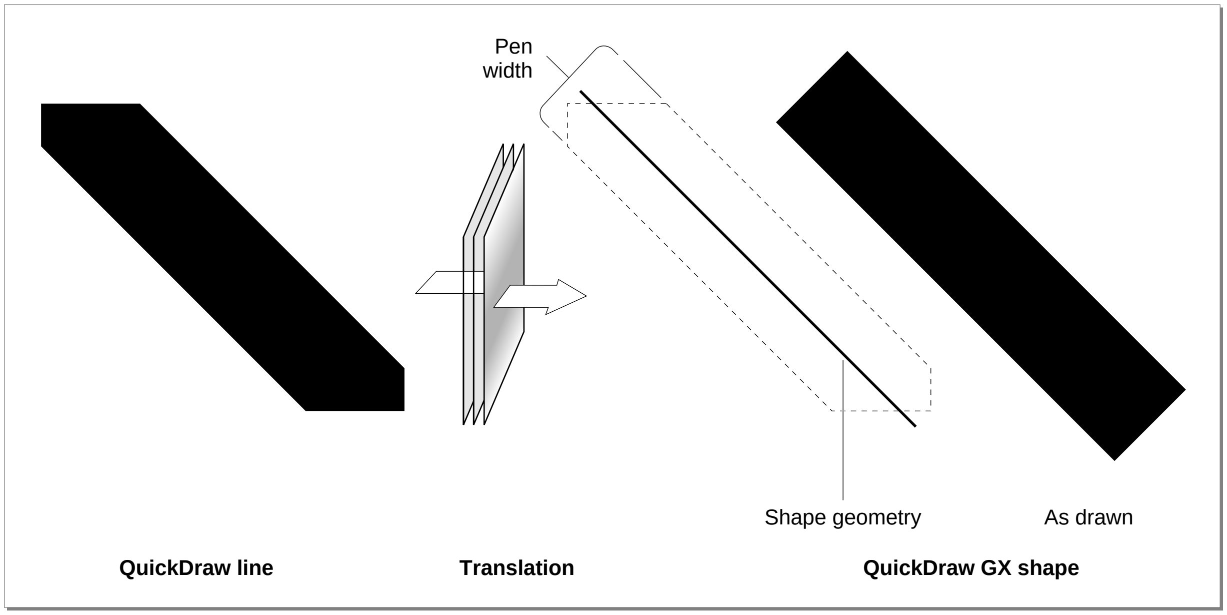
The `gxDefaultOptionsTranslation` setting produces the best replication of the original QuickDraw picture. However, it is also the slowest translation. If you use the `gxDefaultOptionsTranslation` setting for the translation of the original QuickDraw line shown in Figure 1-2, the resulting QuickDraw GX polygon shape mimics the QuickDraw hanging pen. Furthermore, any scaling between the source and destination rectangles is incorporated into the translated shape’s geometry. The original QuickDraw line would be translated to the QuickDraw GX shape shown in Figure 1-3.

Figure 1-3 Translation of the QuickDraw line using `gxDefaultOptionsTranslation`



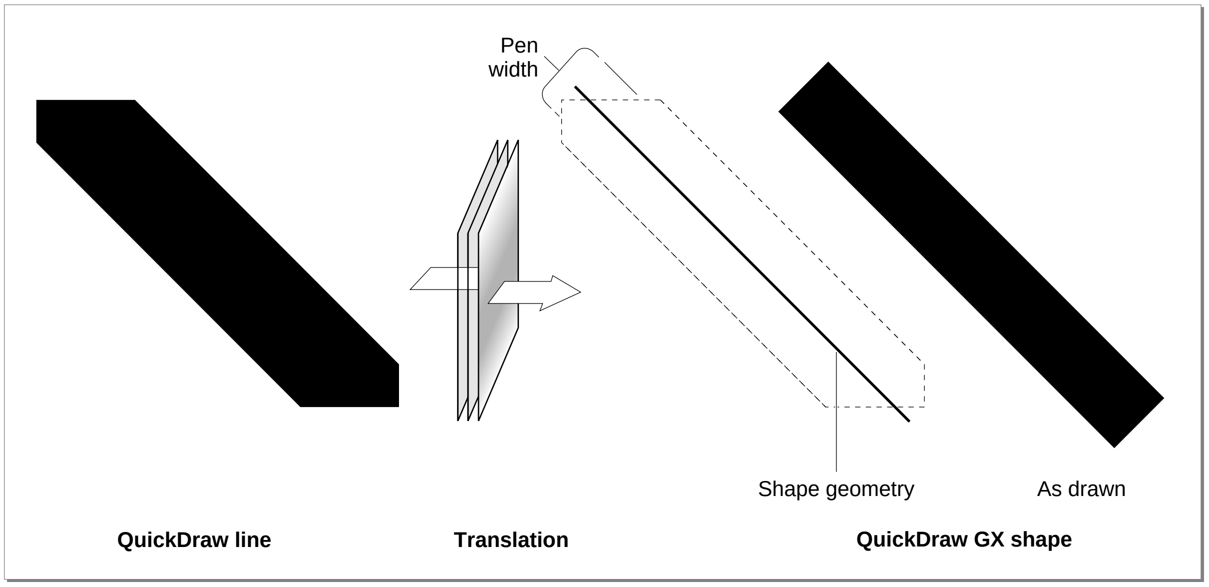
If you use the `gxSimpleGeometryTranslation` option setting, the resulting QuickDraw GX line shape runs along the center of the original QuickDraw line and covers all the pixels of the QuickDraw line and more; it is a superset. The resulting QuickDraw GX shape looks like the line shape shown in Figure 1-4.

Figure 1-4 Translation of the QuickDraw line using `gxSimpleGeometryTranslation`



If you use the `gxReplaceLineWidthTranslation` option setting, the resulting QuickDraw GX line shape has a width that is the average of the QuickDraw pen width and height. The line runs along the center of the original QuickDraw line between the extreme pixels at each end of the original QuickDraw line. The translation results in the QuickDraw GX shape shown in Figure 1-5.

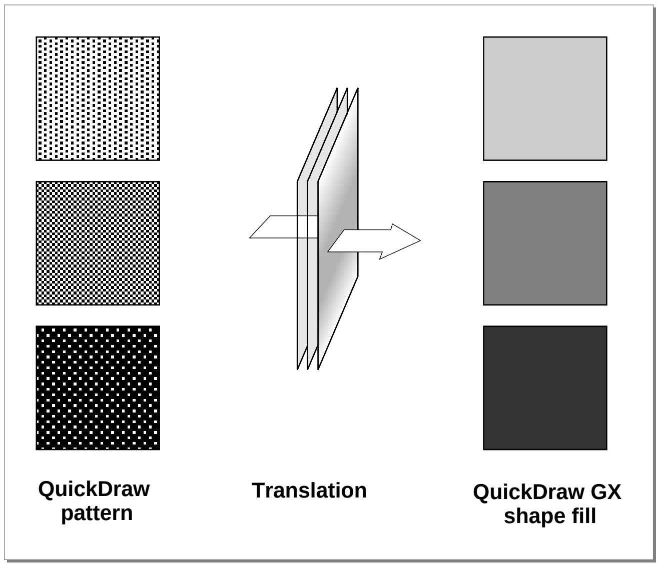
Figure 1-5 Translation of the QuickDraw line using `gxReplaceLineWidthTranslation`



Translation of Fill Patterns

The QuickDraw–to–QuickDraw GX translator converts those 8-bit × 8-bit QuickDraw fill patterns that are commonly used to represent gray patterns to colors that are blends of the foreground and background colors. In the case of QuickDraw black-and-white patterns, a uniform grayscale shade that ranges from 0 to 100 percent black is produced, depending on the overall apparent density of the original pattern, as shown in Figure 1-6.

Figure 1-6 Conversion of standard QuickDraw fill patterns to QuickDraw GX shape fills



Translation of QuickDraw Picture Comments

The capabilities of QuickDraw GX exceed those of QuickDraw. This means that a picture comment (`picComment`) can be incorporated into the translated shapes as part of the conversion process. With QuickDraw alone, picture comments can only be seen when the picture is printed, because the comments are interpreted at the printer level.

It is common practice for developers to include QuickDraw drawing commands (usually one or more **bitmaps**) within a picture comment as a proxy that provides an alternate representation of the picture comment. That way, if the `picComment` is not supported by a printer, some output—although at a lower resolution— is produced.

When processing a picture comment, the QuickDraw–to–QuickDraw GX translator typically discards the QuickDraw proxy and applies the `picComment` to the object—for example, by rotating the shape's transform or setting a dash in the shape's style. When processing PostScript picture comments, however, the translator creates a picture shape that contains QuickDraw GX **shape objects** (based on the QuickDraw proxies) as well as **tag objects** (containing the PostScript data). In this way, QuickDraw GX can render the picture both on a raster device (by drawing the items in the picture shape) and on a PostScript device (by applying the information in the tag objects).

Sample code for applying a `picComment` for rotation is shown in Listing 1-3.

Listing 1-3 QuickDraw picture data that includes a `picComment`

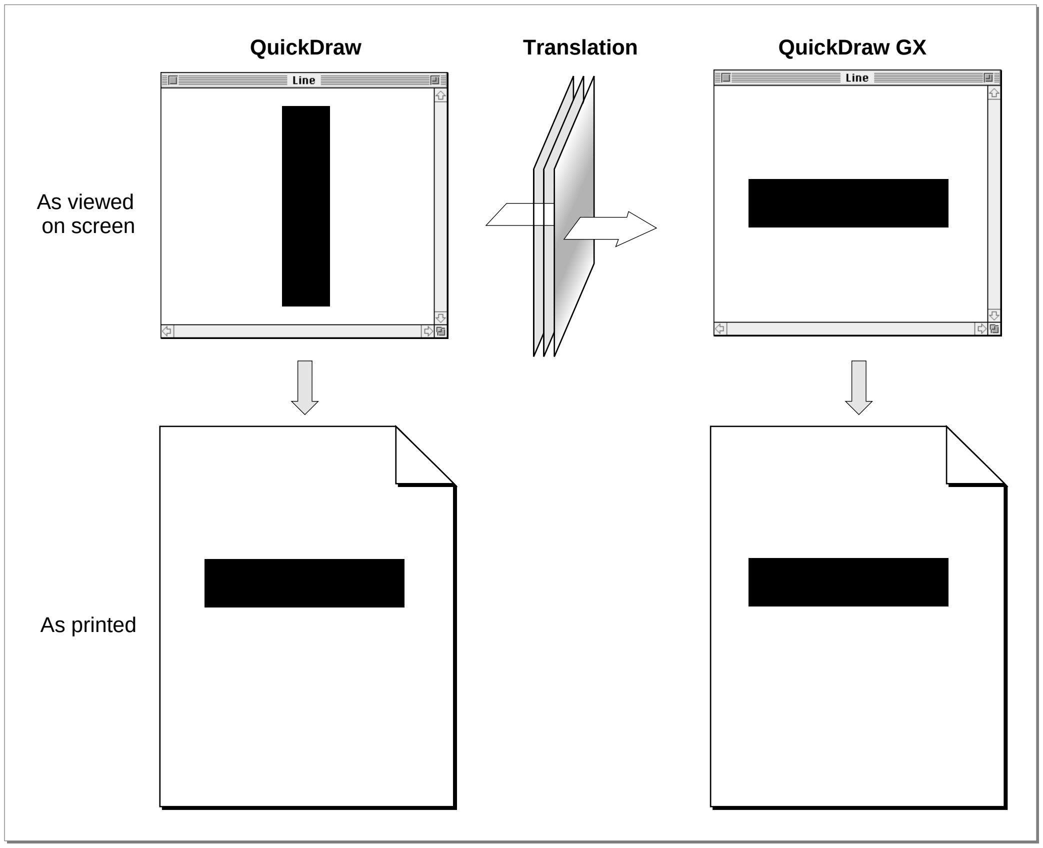
```
RotComHandle   rInfo = NewHandle(sizeof(RotComRecord));

(*rInfo)->rFlip = 0
(*rInfo)->rAngle = 90;

MoveTo(100,100);
PicComment(RotateBegin, sizeof(RotComRecord), (Handle)rInfo);
LineTo(100, 200);
PicComment(RotateEnd, 0, nil);
```

The output of the sample code in Listing 1-3 is shown in Figure 1-7. Notice that the QuickDraw screen output is not rotated. This is because QuickDraw picture comments are interpreted by the printer. In contrast, the printed QuickDraw output and the translated QuickDraw GX shape (both printed and displayed onscreen) correctly represent the intent of the original QuickDraw data.

Figure 1-7 Translating QuickDraw data containing a rotation `picComment`



Translation Statistics

The translator keeps various statistics about the QuickDraw picture data that it translates. You can examine these statistics after the translation if you are interested in this information. The statistics information is returned in the form of bit flags, as shown in Table 1-3.

Table 1-3 Translation statistics options

Constant	Value	Explanation
`gxContainsFormsBegin`	0x0001	The data that was translated contained "`formsBegin`" picture comments.
`gxContainsFormsEnd`	0x0002	The data that was translated contained "`formsEnd`" picture comments.
`gxContainsPostScript`	0x0004	The data that was translated contained PostScript picture comments.
`gxContainsEmptyPostScript`	0x0008	The data that was translated contained PostScript picture comments in which there was no actual PostScript data.

Using the Translator With QuickDraw Pictures

If you have a handle to QuickDraw picture data, such as from a file or on the Clipboard, you can convert that data into a QuickDraw GX picture shape with a single call to the QuickDraw–to–QuickDraw GX translator.

You use the `GXConvertPICTToShape` function to translate an entire QuickDraw picture into a QuickDraw GX shape. You pass the picture handle of the QuickDraw picture you wish to translate and a reference to a shape into which the translated data is to be placed. Listing 1-4 is a sample that uses the `GXConvertPICTToShape` function to perform the translation, and then draws the resultant picture shape to the view port specified in the view port array `thePorts`.

Listing 1-4 Translating QuickDraw picture data with `GXConvertPICTToShape`

```
aPicShape = GXNewShape(gxPictureType);

GXConvertPICTToShape(thePicHdl, gxDefaultOptionsTranslation,
                     &theRect, &theRect, styleStretch,
                     aPicShape, nil);

GXSetShapeViewPorts(aPicShape,1,thePorts);
GXDrawShape(aPicShape);
GXDisposeShape(aPicShape);
```

Installing and Removing the Translator

If you want to capture QuickDraw commands as they are executed, convert them, and either draw them immediately or save them, you need to install the QuickDraw–to–QuickDraw GX translator in the graphics port to which the QuickDraw drawing commands will be sent.

You install the translator with the `GXInstallQDTranslator` function. Once installed, the translator intercepts all QuickDraw drawing commands to that port, converts them to QuickDraw GX shapes, and sends them to a callback function (that you supply) for drawing or saving. When you are finished capturing QuickDraw commands, you remove the translator with the `GXRemoveQDTranslator` function.

Note
There is not necessarily a one-to-one match between a QuickDraw function call and the generation of a QuickDraw GX shape. ◆

Listing 1-5 is a sample that uses the functions `GXInstallQDTranslator` and `GXRemoveQDTranslator` to convert the bounded QuickDraw commands. The application-defined callback function, `aShapeProc`, sets the view port and draws each translated shape. The `aShapeProc` function is shown in Listing 1-6 on page 1-22.

Listing 1-5 Installing and removing the translator

```
/* first, install the translator */
GXInstallQDTranslator(window, gxDefaultOptionsTranslation,
                        &theRect, &theRect, theStyleStretch,
                        aShapeProc, (void *) &theWindViewPort);

/* now, make QuickDraw calls */
PenSize(20, 10);

MoveTo(100, 100);
LineTo(200, 100);

MoveTo(100, 150);
LineTo(200, 250);

/* when finished drawing, remove the translator */
GXRemoveQDTranslator(window, nil);
```

When using the `GXInstallQDTranslator` function, you must supply an application-defined function that gives you control over what is to be done with the QuickDraw GX shapes resulting from the translation. For example, you may want to draw each shape as it is translated, or you may want to spool multiple shapes and draw after you have completed the picture.

Listing 1-6 is the sample shape-spooling function used by the code in Listing 1-5. This function sets the view port, which is passed to it in the `reference` parameter, and then draws the shape passed to it in the parameter `theShape`.

Listing 1-6 Sample application-defined shape-spooling function

```
OSErr aShapeProc( gxShape theShape,
                                 void *reference)
{
   GXSetShapeViewPorts(theShape, 1, (gxViewPort *) &reference);
   GXDrawShape(theShape);
   GXDisposeShape(theShape);
   return(GXGetGraphicsError(nil));
}
```

The prototype for the shape-spooling function, and how to use it, are described in the section "Handling Translated QuickDraw Data" beginning on page 1-41.

QuickDraw GX and the Macintosh Environment Reference

This section contains constants, data types, and functions that are specific to the QuickDraw GX environment.

Constants and Data Types

This section describes the constants that you can use with the `Gestalt` function and the constants you can use to control translation with the QuickDraw–to–QuickDraw GX translator.

Gestalt Selectors and Attributes

The selector `'grfx'` can be used with the `Gestalt` function to determine whether the graphics and typography portions of QuickDraw GX have been installed. The `'pmgr'` selector can be used to determine whether QuickDraw GX printing is installed. `Gestalt` returns the version number in either case.

If you call `Gestalt` with the `gestaltGraphicsAttr` selector, it returns an attribute that specifies whether the debugging or nondebugging version of QuickDraw GX is installed, and what platform it is installed on. You can use the `'qdgx'` selector to determine if QuickDraw GX is installed.

```
#define gestaltGXVersion                'qdgx'
#define gestaltGraphicsVersion          'grfx'
#define gestaltPrintingMgrVersion       'pmgr'
#define gestaltCurrentGraphicsVersion   0x00010000

#define gestaltGraphicsAttr             'gfxa'
#define gestaltGraphicsisDebugging      0x00000001
#define gestaltGraphicsisLoaded         0x00000002
#define gestaltGraphicsIsPowerPC        0x00000004
```

These selectors and attributes are described in the section "Testing for the Presence and Version of QuickDraw GX" beginning on page 1-4. The `Gestalt` function is described in the chapter "Gestalt Manager" in *Inside Macintosh: Operating System Utilities.*

Translator Options and Statistics

The `gxTranslationOptions` enumeration defines constants that control various aspects of the translation from QuickDraw to QuickDraw GX:

```
enum gxTranslationOptions {
   gxDefaultOptionsTranslation   = 0x0000,
   gxOptimizedTranslation        = 0x0001,
   gxReplaceLineWidthTranslation = 0x0002,
   gxSimpleScalingTranslation    = 0x0004,
   gxSimpleGeometryTranslation   = 0x0008,
   gxSimpleLinesTranslation      = 0x000C,
   gxLayoutTextTranslation       = 0x0010,
   gxRasterTargetTranslation     = 0x0020,
   gxPostScriptTargetTranslation = 0x0040,
   gxVectorTargetTranslation     = 0x0080
};
typedef long gxTranslationOption;
```

The individual constants for the enumeration are described in Table 1-2 on page 1-12.

The `gxTranslationStatistics` enumeration defines constants that are used as masks, any of which you can combine using an AND operation to interpret the statistics gathered during translation:

```
enum gxTranslationStatistics {
   gxContainsFormsBegin       = 0x0001,
   gxContainsFormsEnd         = 0x0002,
   gxContainsPostScript       = 0x0004,
   gxContainsEmptyPostScript  = 0x0008
};
typedef long gxTranslationStatistic;
```

The individual constants for the enumeration are described in Table 1-3 on page 1-20.

Macintosh Interface Functions

This section describes the QuickDraw GX functions you can use to

- associate view ports with Macintosh windows
- associate view devices with Macintosh graphics devices (`GDevice` records)
- convert QuickDraw coordinates and mouse locations to QuickDraw GX coordinates
- install and remove view port filters that intercept QuickDraw GX drawing commands

Associating View Ports With Macintosh Windows

This section describes the function you use to

- create a new view port object associated with a specific Macintosh window
- retrieve the Macintosh window associated with a view port, or the view port associated with a Macintosh window

GXNewWindowViewPort

You can use the `GXNewWindowViewPort` function to create a new view port for a specified Macintosh window.

```
gxViewPort GXNewWindowViewPort(WindowPtr qdWindow);
```

`qdWindow`	A pointer to the window for which the new view port is to be created.
function result	A reference to the new view port.

DESCRIPTION

The `GXNewWindowViewPort` function creates a new view port associated with the specified window. All drawing in the window view port will be clipped to the visible region of the window.

View ports associated with windows are clipped by the visible region (`visRgn`), but not the clip region (`clipRgn`) of the window. The origin of the window doesn't affect the view port. The clip shape of the view port doesn't affect drawing in the window.

SPECIAL CONSIDERATIONS

You cannot alter the mapping or clip properties of view ports created with this function. Most typically, you use this function to create a view port attached to a window, and then—if you support scrolling or otherwise need to change the clip or mapping—you create one or more child view ports of the window view port and draw into them.

Do not attach more than one view port to a window through this function; unpredictable behavior results.

ERRORS, WARNINGS, AND NOTICES

Errors
`out_of_memory`

SEE ALSO

View ports, child view ports, and the limitations on access to window view ports are discussed in the chapter "View-Related Objects" in *Inside Macintosh: QuickDraw GX Objects.*

To obtain the window associated with a view port, use the `GXGetViewPortWindow` function, described next. To obtain the view port associated with a window, use the `GXGetWindowViewPort` function, described on page 1-26.

GXGetViewPortWindow

You can use the `GXGetViewPortWindow` function to return the Macintosh window of a specified view port.

```
gxWindowPtr GXGetViewPortWindow(gxViewPort portOrder);
```

`portOrder`	A reference to the specified view port.
function result	A pointer to the window associated with the specified view port.

DESCRIPTION

The function returns `nil` if the view port is not associated with a window.

This function returns `nil` if you pass it a reference to a child view port of a window view port. To determine the window ultimately associated with a child view port, use the `GXGetViewPortParent` function to find the parent view port at the top of the view port hierarchy, and pass that view port reference to the `GXGetViewPortWindow` function.

ERRORS, WARNINGS, AND NOTICES

Errors
`out_of_memory`
`invalid_viewport_reference`

SEE ALSO

To create a view port associated with a window, use the `GXNewWindowViewPort` function, described in the previous section. To obtain the view port associated with a window, use the `GXGetWindowViewPort` function, described next.

GXGetWindowViewPort

You can use the `GXGetWindowViewPort` function to return the view port of a specified Macintosh window.

```
gxViewPort GXGetWindowViewPort(WindowPtr qdWindow);
```

`qdWindow`	A pointer to the specified window.
function result	A reference to the view port associated with the specified window.

DESCRIPTION

The function returns `nil` if the window has no associated view port.

ERRORS, WARNINGS, AND NOTICES

Errors
`out_of_memory`

SEE ALSO

To create a view port associated with a window, use the `GXNewWindowViewPort` function, described on page 1-24. To obtain the window associated with a view port, use the `GXGetViewPortWindow` function, described in the previous section.

Associating View Devices With Macintosh Graphics Devices

This section describes the functions you use to retrieve the Macintosh graphics device (`GDevice` record) associated with a QuickDraw GX view device object, or the view device associated with a Macintosh graphics device.

GXGetViewDeviceGDevice

You can use the `GXGetViewDeviceGDevice` function to return the Macintosh graphics device associated with a specified view device object.

```
GDHandle GXGetViewDeviceGDevice(gxViewDevice theDevice);
```

`theDevice`	A reference to the view device whose graphics device is requested.
function result	A handle to the `GDevice` record of the specified view device.

DESCRIPTION

The `GXGetViewDeviceGDevice` function returns a handle to the `GDevice` record associated with a specified view device. The function returns `nil` if the view device has no `GDevice` record.

ERRORS, WARNINGS, AND NOTICES

Errors
```
out_of_memory
invalid_viewdevice_reference
```

SEE ALSO

Macintosh graphics devices and the `GDevice` record are described in *Inside Macintosh: Imaging With QuickDraw.*

View devices are described in the chapter “View-Related Objects” in *Inside Macintosh: QuickD*raw GX Objects.

To obtain the view device associated with a graphics device, use the `GXGetGDeviceViewDevice` function, described next.

GXGetGDeviceViewDevice

You can use the `GXGetGDeviceViewDevice` function to return the view device object associated with a specified Macintosh graphics device.

```
gxViewDevice GXGetGDeviceViewDevice(GDHandle qdGDevice);
```

`qdGDevice`	A handle to the `GDevice` record of the graphics device whose view device is requested.
function result	A reference to the view device object associated with the specified graphics device.

DESCRIPTION

The `GXGetGDeviceViewDevice` function returns a reference to the view device object associated with a specified graphics device. The function returns `nil` if the graphics device has no view device.

ERRORS, WARNINGS, AND NOTICES

Errors
`out_of_memory`

SEE ALSO

Macintosh graphics devices and the `GDevice` record are described in *Inside Macintosh: Imaging With QuickDraw.*

View devices are described in the chapter "View-Related Objects" in *Inside Macintosh: QuickD*raw GX Objects.

To obtain the graphics device associated with a view device, use the `GXGetViewDeviceGDevice` function, described in the previous section.

Converting From QuickDraw to QuickDraw GX Coordinates

This section describes the functions you use to

- convert from QuickDraw coordinates to QuickDraw GX coordinates
- retrieve mouse locations in terms of QuickDraw GX coordinates

GXConvertQDPoint

You can use the `GXConvertQDPoint` function to convert from QuickDraw global coordinates to QuickDraw GX coordinates.

```
void GXConvertQDPoint(const Point *shortPt, gxViewPort portOrder,
                      gxPoint *fixedPt);
```

`shortPt`	A pointer to a point in QuickDraw global coordinates that is to be converted to QuickDraw GX coordinate space.
`portOrder`	A reference to a view port. If this parameter is `nil`, the conversion is to QuickDraw GX global coordinates; if it is other than `nil`, the conversion is to the local coordinates of that view port.
`fixedPt`	A pointer to a `gxPoint` structure. On return, the structure contains the result of the coordinate conversion.

DESCRIPTION

The `GXConvertQDPoint` function converts a point with QuickDraw global coordinates to a point with QuickDraw GX coordinates. If the `portOrder` parameter is `nil`, the QuickDraw global coordinates are converted to QuickDraw GX global coordinates. If the `portOrder` parameter is specified, the QuickDraw global coordinates are converted to QuickDraw GX local coordinates. The local coordinates are local within the specified view port.

The QuickDraw global coordinates are specified in pixels. The QuickDraw GX global and local coordinates are specified in a coordinate space in which 1.0 = 1/72 inch.

ERRORS, WARNINGS, AND NOTICES

Errors
`out_of_memory`
`invalid_viewPort_reference`

Warnings
`point_does_not_intersect_port` (debugging version)

SEE ALSO

For more information about converting from QuickDraw to QuickDraw GX coordinate space, see the section “Converting From QuickDraw to QuickDraw GX Coordinates” beginning on page 1-7. For more information about the QuickDraw GX coordinate system and drawing, see the chapter “View-Related Objects” in *Inside Macintosh: QuickDraw GX Objects.*

GXGetGlobalMouse

You can use the `GXGetGlobalMouse` function to obtain the current cursor position in QuickDraw GX global coordinates.

```
void GXGetGlobalMouse(gxPoint *globalPt);
```

`globalPt`	A pointer to a `gxPoint` structure. On return, the structure contains the fixed-point global coordinates of the current cursor position.

ERRORS, WARNINGS, AND NOTICES

Errors
`out_of_memory`

GXGetViewPortMouse

You can use the `GXGetViewPortMouse` function to obtain the cursor position expressed in the QuickDraw GX local coordinates of the specified view port.

```
void GXGetViewPortMouse(gxViewPort portOrder, gxPoint *localPt);
```

`portOrder`	A reference to the view port for which the cursor position is requested.
`localPt`	A pointer to a `gxPoint` structure. On return, the structure contains the fixed-point local coordinates of the current cursor position.

SPECIAL CONSIDERATIONS

If the `portOrder` parameter is `nil`, this function posts an `invalid_viewPort_reference` error.

ERRORS, WARNINGS, AND NOTICES

Errors
`out_of_memory`
`invalid_viewPort_reference`

Warnings
`point_does_not_intersect_port`

Installing a View Port Filter

This section describes the functions you use to install and remove view port filters, which allow you to intercept drawing commands.

GXSetViewPortFilter

You can use the `GXSetViewPortFilter` function to intercept a shape being sent to a specified view port. Instead of drawing the shape, you can use an application-defined filter function to manipulate the shape.

```
void GXSetViewPortFilter(gxViewPort portOrder,
                        gxUserViewPortFilter filter,long refCon);
```

`portOrder`	A reference to the view port to be filtered.
`filter`	A pointer to an application-defined callback function that acts on the filtered shape. If you pass `nil` for this parameter, any installed filter function is removed.
`refCon`	A `long` value to be passed to the callback filter function. Your filter function can use the value in any manner.

DESCRIPTION

The `GXSetViewPortFilter` function allows you to install a filter function that intercepts shapes sent to a specified view port and manipulates them in any manner you wish. You must specify a valid view port reference in the `portOrder` parameter.

The filter parameter is a pointer to an application-defined callback function of type `gxUserViewPortFilter`:

```
typedef void (*gxUserViewPortFilterProcPtr)(gxShape toFilter,
                              gxViewPort portOrder, long refCon);
typedef gxUserViewPortFilterProcPtr gxUserViewPortFilter;
```

You must provide the callback filter function; its prototype is described in the section “Filtering Drawing Calls to a View Port” beginning on page 1-40.

When you call `GXSetViewPortFilter`, you can pass it any useful `long` value in the `refCon` parameter. That value will be passed to the filter function each time it is called.

To remove a filter function from a view port, call `GXSetViewPortFilter` with a `nil` value for the `filter` parameter.

SPECIAL CONSIDERATIONS

If you assign a filter function to a view port, it affects drawing to that specific view port only. Drawing to its child view ports or its parent view port (or any other view ports within its hierarchy) is unaffected.

ERRORS, WARNINGS, AND NOTICES

Errors
`out_of_memory`
`invalid_viewPort_reference`

SEE ALSO

To obtain a pointer to the filter function currently installed in a given view port, use the `GXGetViewPortFilter` function, described next.

For a description of the application-defined view port filter function, see page 1-40.

GXGetViewPortFilter

You can use the `GXGetViewPortFilter` function to return the view device of a specified graphics device.

```
gxUserViewPortFilter GXGetViewPortFilter(gxViewPort portOrder,
                                         long *refCon);
```

`portOrder`	A reference to the view port whose currently installed view port filter you need.
`refCon`	A pointer to a `long` value. On return, the value is the reference constant that was passed to the `GXSetViewPortFilter` function when the filter function was installed.
function result	A pointer to the view port filter function that is installed in the specified view port.

DESCRIPTION

The `GXGetViewPortFilter` function returns a pointer to the view port filter that is currently installed in the specified view port. The function also returns, in the `refCon` parameter, the reference constant that was passed to `GXSetViewPortFilter` when the filter function was installed.

ERRORS, WARNINGS, AND NOTICES

Errors
`out_of_memory`
`invalid_viewPort_reference`

SEE ALSO

The `GXSetViewPortFilter` function is described in the previous section.

For a description of the prototype of the view port filter function, see page 1-40.

QuickDraw–to–QuickDraw GX Translator Functions

This section describes the functions you can use to

- Convert the specification for a font and face in the GrafPort into a gxStyle
- Convert QuickDraw picture data into QuickDraw GX shapes
- Convert QuickDraw commands to QuickDraw GX shapes

Converting a GrafPort Font and Face Specification

You use the function described in this section to convert the specification for a font and face in the GrafPort into a gxStyle.

GXConvertQDFont

You use the `GXConvertQDfont` function to translate the specification for a font and face in the GrafPort into a gxStyle.

```
long GXConvertQDfont(gxStyle theStyle, long txFont, long txFace);
```

`theStyle`	A gxStyle.
`txFont`	A long specifying a text font; same as in the GrafPort.
`txFace`	A long specifying a text style; same as in the GrafPort.

DESCRIPTION

The `GXConvertQDfont` function picks the `gxFont` that is the closest match for the `txFont` and `txFace` parameters. If it does not find an exact match, `GXConvertQDfont` might also set the style's font variation.

The `GXConvertQDfont` function also sets the style's encoding. It returns any style bits from `txFace` that were not accounted for in the `gxFont` and variation. This permits the caller to construct a `gxTextFace` based on the returned style bits. Potentially all of the style bits can be matched. Currently only bold, italic, condense, and extended bits are matched. However, in the future more might be matched. You should not make any assumptions about what will or will not be matched.

SPECIAL CONSIDERATIONS

If the translator calls `GXConvertQDFont`, it will already have mapped `txFont==0` to the correct font by calling the Script Manager.

Converting QuickDraw Pictures

You use the function described in this section to convert QuickDraw picture data to a QuickDraw GX picture shape.

GXConvertPICTToShape

You can use the `GXConvertPICTToShape` function to convert a QuickDraw picture to a QuickDraw GX shape.

```
gxShape GXConvertPICTToShape(const PicHandle pict,
                        gxTranslationOptions options,
                        const Rect *srcRect,
                        const Rect *dstRect,
                        Point styleStretch,
                        gxShape destination,
                        gxTranslationStatistic *statistics);
```

`pict`	A handle to the QuickDraw picture image to be converted to QuickDraw GX.
`options`	The translation options to use for the translation.
`srcRect`	A pointer to the source rectangle (normally the QuickDraw picture frame) of the QuickDraw image, in QuickDraw coordinates.
`dstRect`	A pointer to the destination rectangle of the QuickDraw image, in QuickDraw coordinates.
`styleStretch`	The scale factor (both horizontal and vertical) to apply to certain items, such as dashes, in QuickDraw picture comments.

`destination`	A reference to the destination shape—the shape in which to store the translated picture.
`statistics`	A pointer to the location in which to store the translation statistics.
function result	A reference to the destination shape. This is the same shape referenced in the `destination` parameter, or a newly created picture shape if the `destination` parameter was `nil`.

DESCRIPTION

The `GXConvertPICTToShape` function provides a conversion of a QuickDraw GX picture image within the boundaries of the source rectangle `srcRect` to a QuickDraw GX shape within the boundaries of the destination rectangle `dstRect`. You pass QuickDraw picture to the translator and it returns a QuickDraw GX picture shape that approximates the original QuickDraw picture.

The `srcRect` and `dstRect` parameters represent the source and destination space of the image, expressed in QuickDraw coordinates. The relation between the source and destination rectangles specifies the amount of scaling to be applied during translation. You can use this scaling capability to ensure that the resulting QuickDraw GX shape is a good representation of the original QuickDraw object. The `srcRect` parameter is normally the QuickDraw picture frame of the original picture data. If scaling is not required, you can pass in the source rectangle of the original data for both parameters.

If the `destination` parameter is `nil`, a QuickDraw GX picture shape is created and returned by the function. If the destination parameter is not `nil`, the function returns the same shape reference that it was passed. The QuickDraw GX shapes resulting from the translation are contained in that picture shape.

The translator keeps various statistics about the QuickDraw picture data it is translating. You can examine the mask that describes these statistics after the translation if you are interested in this information. The flags in the translation statistics masks are defined in the `gxTranslationStatistics` enumeration. If you are not interested in this information, you should pass `nil` for the `stats` parameter.

SPECIAL CONSIDERATIONS

If you pass `nil` for the `destination` parameter and if no error occurs, this function creates a QuickDraw GX shape object. You are responsible for disposing of that object when you no longer need it.

Before using the translator, you must first call the `GXInitPrinting` function.

ERRORS, WARNINGS, AND NOTICES

Errors
`out_of_memory`

SEE ALSO

For an example of the use of this function, see Listing 1-4 on page 1-20.

To translate individual QuickDraw drawing commands as they are executed, use the `GXInstallQDTranslator` function, described next.

Translation options from the `gxTranslationOptions` enumeration are described in Table 1-2 on page 1-12.

Translation statistics flags from the `gxTranslationStatistics` enumeration are described in Table 1-3 on page 1-20.

For a general description of the QuickDraw–to–QuickDraw GX translator, see the section "Using the QuickDraw–to–QuickDraw GX Translator" beginning on page 1-10.

The `GXInitPrinting` function is described in the chapter "Core Printing Features" in *Inside Macintosh: QuickDraw GX Printing*.

Installing and Removing the Translator

This section describes the functions you use to install the QuickDraw–to–QuickDraw GX translator in order to intercept QuickDraw drawing commands as they are executed. To use the translator in this way, you also need to supply a callback shape-spooling function, described on page 1-41.

GXInstallQDTranslator

You can use the `GXInstallQDTranslator` function to initiate translation of the QuickDraw drawing commands to QuickDraw GX shapes. Subsequent QuickDraw drawing commands are translated into equivalent QuickDraw GX shapes.

```
void GXInstallQDTranslator(GrafPtr port,
                           gxTranslationOptions options,
                           const Rect *srcRect,
                           const Rect *dstRect,
                           Point styleStretch,
                           gxShapeSpoolFunction userFunction,
                           void *reference);
```

`port`	A pointer to the QuickDraw graphics port into which to install the translator.
`options`	The translation options to use for the translation.

`srcRect`	A pointer to a rectangle defining the QuickDraw source dimensions for drawing, in QuickDraw coordinates.
`dstRect`	A pointer to a rectangle defining the QuickDraw destination dimensions for drawing, in QuickDraw coordinates.
`styleStretch`	The amount of scaling that certain picture-comment items, such as dashes, will be given in the x and y directions.
`userFunction`	A pointer to an application-defined callback function to which the translator passes each translated shape.
`reference`	A pointer to a reference constant that can be used for any purpose. QuickDraw GX passes the reference constant to the application-defined callback function each time it calls the function.

DESCRIPTION

You can use the `GXInstallQDTranslator` function to install the QuickDraw–to–QuickDraw GX translator into a QuickDraw graphics port. QuickDraw commands that draw into that port are translated to equivalent QuickDraw GX shapes.

All QuickDraw drawing commands executed after you call this function and before you call `GXRemoveQDTranslator` are converted into QuickDraw GX shapes and passed to an application-defined function. There is not necessarily a one-to-one match between a QuickDraw function call and the generation of a QuickDraw GX shape; the translation algorithm can combine several QuickDraw items into one QuickDraw GX shape.

The `srcRect` and `dstRect` parameters represent the source and destination space of the image, expressed in QuickDraw coordinates. The relation between the source and destination rectangles specifies the amount of scaling to be applied during translation. If scaling is not required, you can pass identical rectangles for both parameters.

The `styleStretch` parameter represents the amount of scaling that QuickDraw bitmap patterns are given by the translator in the x and y directions in order to scale them up to the destination space. The x scale factor is stored in `styleStretch.h` and the y scale factor is stored in `styleStretch.v`. These values are usually the destination resolution divided by the screen resolution (72 dpi), rounded to the nearest integer.

The `userFunction` parameter is a pointer to an application-defined callback function of type `gxShapeSpoolFunction`:

```
typedef OSErr (*gxShapeSpoolProcPtr)(gxShape toSpool,
                                              void *refCon);
typedef gxShapeSpoolProcPtr gxShapeSpoolFunction;
```

The translator calls the function every time that it generates a translated QuickDraw GX shape. You must provide the function; its prototype is described in the section “Handling Translated QuickDraw Data” beginning on page 1-41.

The `reference` parameter of the `GXInstallQDTranslator` function can be used by your application in any manner you desire, within the constraints of a `long` data field.

The translator passes the parameter to the application-defined callback function. For example, you can use `reference` to specify where the translated picture is to be displayed, by passing a view port reference in the `reference` parameter.

SPECIAL CONSIDERATIONS

If you call the `GXInstallQDTranslator` function to install the translator, you must subsequently call the `GXRemoveQDTranslator` function to remove it when you are finished.

The `port` parameter to this function must be an old-style graphics-port pointer (`GrafPtr`); however, the translator depends on the existence of a color graphics port. Therefore, you should always create a color graphics port (using `NewCWindow`, `NewGWorld` or `OpenCPort`), and coerce the `CGrafPtr` to a `GrafPtr` when you call this function.

Before installing the translator, you must first call the `GXInitPrinting` function.

Installation of the translator cannot cause an `out_of_memory` error, but the process of translation can result in an out-of-memory condition. Your application should be prepared to handle that condition.

ERRORS, WARNINGS, AND NOTICES

Notices (debugging version)
`translator_already_installed_on_this_grafport`

SEE ALSO

For an example of the use of this function, see Listing 1-5 on page 1-21.

Translation options from the `gxTranslationOptions` enumeration are described in Table 1-2 on page 1-12.

The application-defined shape-spooling function called by the translator is described on page 1-41.

To translate an entire QuickDraw picture, use the `GXConvertPICTToShape` function, described in the previous section.

For a general description of the QuickDraw–to–QuickDraw GX translator, see the section “Using the QuickDraw–to–QuickDraw GX Translator” beginning on page 1-10.

The `GXInitPrinting` function is described in the chapter “Core Printing Features” in *Inside Macintosh: QuickDraw GX Printing*.

For descriptions of color graphics ports and the functions you use to create them, see *Inside Macintosh: Imaging With QuickDraw*.

GXRemoveQDTranslator

You can use the `GXRemoveQDTranslator` function to terminate the translation of QuickDraw drawing commands.

```
translationStatistics GXRemoveQDTranslator(GrafPtr port,
                             gxTranslationStatistic *statistic);
```

`port`	A pointer to the QuickDraw graphics port in which the translator is currently installed.
`statistic`	A pointer to the location in which to return the translation statistics.

function result	The translation statistics.

DESCRIPTION

The `GXRemoveQDTranslator` function removes the translator from the QuickDraw graphics port in which it was installed, and flushes the internal translation buffer.

The translator keeps various statistics about the QuickDraw picture data it is translating. After removing the translator, you can examine the mask that describes these statistics if you are interested in this information. The flags in the translation statistics masks are defined in the `gxTranslationStatistics` enumeration. If you are not interested in this information, you should pass `nil` for the statistic parameter.

SPECIAL CONSIDERATIONS

Always be sure to call the `GXRemoveQDTranslator` function when you are finished translating.

EERRORS, WARNINGS, AND NOTICES

Warnings

`translator_not_installed_on_this_grafport`	(debugging version)

SEE ALSO

For an example of the use of this function, see Listing 1-5 on page 1-21.

Translation statistics flags from the `gxTranslationStatistics` enumeration are described in Table 1-3 on page 1-20.

To translate an entire QuickDraw picture, use the `GXConvertPICTToShape` function, described on page 1-34.

For a general description of the QuickDraw–to–QuickDraw GX translator, see the section “Using the QuickDraw–to–QuickDraw GX Translator” beginning on page 1-10.

Application-Defined Functions

This section describes callback functions that your application must provide for QuickDraw GX to call, in two situations:

- If you intercept shape-drawing calls to a view port
- If you install the QuickDraw–to–QuickDraw GX translator in a graphics port

Filtering Drawing Calls to a View Port

The callback function described in this section is a view port filter, which handles shape-drawing calls that have been intercepted in a given view port.

MyViewPortFilter

You can create a filter function that handles intercepted QuickDraw GX drawing calls. The filter function must have a prototype of this form:

```
void MyViewPortFilter(gxShape toFilter, gxViewPort portOrder,
                     long refCon);
```

`toFilter`	A reference to the shape to be filtered—that is, the shape that would have been drawn to the view port specified in the `portOrder` parameter.
`portOrder`	A reference to the view port in which this filter function has been installed.
`refCon`	A reference constant that your filter function can use in any manner.

DESCRIPTION

Once this filter function is installed, QuickDraw GX calls it any time a function is executed that draws a shape in the view port referenced by the `portOrder` parameter. Instead of drawing the shape, QuickDraw GX passes the shape to this function. Your filter function can perform actions other than drawing (such as spooling), or it can otherwise modify or process the shape.

Your application installs this filter function by providing a pointer to it when calling the `GXSetViewPortFilter` function. When your application calls the function `GXSetViewPortFilter`, it also provides the `refCon` value that will be passed to this filter function.

The value passed to you in the `refCon` parameter can be used for any purpose; for example, it might contain a reference to a different view port for drawing to, a pointer to a buffer for collecting spooled data, or anything else useful to the filter function. The `portOrder` parameter allows you to identify the view port being intercepted, in case the filter function is installed on more than one view port.

Your application can get a pointer to this installed filter function at any time by calling the `GXGetViewPortFilter` function. After you are finished intercepting drawing calls, your application can remove the filter function by calling the `GXSetViewPortFilter` function with a `nil` filter-function pointer.

SEE ALSO

The `GXSetViewPortFilter` function is described on page 1-31. The to `GXGetViewPortFilter` function is described on page 1-32.

Handling Translated QuickDraw Data

The callback function described in this section is a shape-spooling function, which handles QuickDraw GX shapes that have been translated from QuickDraw drawing commands.

MyShapeSpooler

You can create a shape-spooling function that handles QuickDraw drawing commands that have been translated into QuickDraw GX shapes. The shape-spooling function must have a prototype of this form:

```
OSErr MyShapeSpooler(gxShape toSpool, void *refCon);
```

`toSpool`	A reference to the shape just translated by the translator.
`refCon`	A pointer to a reference constant, passed to your spool function by the translator, that you can use for any purpose.
function result	An result code of type `OSErr`. You should pass 0 if no error occurs.

DESCRIPTION

When you install the QuickDraw–to–QuickDraw GX translator with the `GXInstallQDTranslator` function, the translator intercepts all subsequent QuickDraw drawing commands, converts them to QuickDraw GX shapes, and passes those shapes to this shape-spooling function. Your shape-spooling function can draw each shape, add it to a picture, or perform any other action you wish.

You install this function by providing a pointer to it when you call the `GXInstallQDTranslator` function. When you call `GXInstallQDTranslator`, you also provide the `refCon` value that will be passed to this filter function. The `refCon` value can be used for any purpose; for example, it might contain a view port reference for drawing, a pointer to a buffer for collecting spooled data, or anything else useful to the shape-spooling function.

After your application has finished intercepting and converting QuickDraw calls and passing them to this shape-spooling function, your application must remove the translator by calling the `GXRemoveQDTranslator` function.

If your shape-spooling function encounters an error during processing, it should return a nonzero value (usually the error code). If the shape-spooling function returns a nonzero value, the translator ceases translating QuickDraw commands. If an error occurs, your application must still call the `GXRemoveQDTranslator` function to remove the translator.

SEE ALSO

The `GXInstallQDTranslator` function is described on page 1-36. The `GXRemoveQDTranslator` function is described on page 1-39.

For a general description of the QuickDraw–to–QuickDraw GX translator, see the section "Using the QuickDraw–to–QuickDraw GX Translator" beginning on page 1-10.

Summary of QuickDraw GX and the Macintosh Environment

Constants and Data Types

Gestalt Selectors and Attributes

```
#define gestaltGXVersion             'qdgx'   /* Overall GX vers. selector */
#define gestaltGraphicsVersion       'grfx'   /* GX graphics vers. selector */
#define gestaltPrintingMgrVersion    'pmgr'   /* GX printing vers. selector */
#define gestaltCurrentGraphicsVersion  0x00010000           /* version 1.0 */

#define gestaltGraphicsAttr          'gfxa'       /* GX attributes selector */
#define gestaltGraphicsisDebugging  0x00000001
#define gestaltGraphicsisLoaded     0x00000002
#define gestaltGraphicsIsPowerPC    0x00000004
```

Translator Options and Statistics

```
   enum gxTranslationOptions {
      gxDefaultOptionsTranslation   = 0x0000,
      gxOptimizedTranslation        = 0x0001,
      gxReplaceLineWidthTranslation = 0x0002,
      gxSimpleScalingTranslation    = 0x0004,
      gxSimpleGeometryTranslation   = 0x0008,
      gxSimpleLinesTranslation      = 0x000C,
      gxLayoutTextTranslation       = 0x0010,
      gxRasterTargetTranslation     = 0x0020,
      gxPostScriptTargetTranslation = 0x0040,
      gxVectorTargetTranslation     = 0x0080
   };
   typedef long gxTranslationOption;

   enum gxTranslationStatistics {
      gxContainsFormsBegin       = 0x0001,
      gxContainsFormsEnd         = 0x0002,
      gxContainsPostScript       = 0x0004,
      gxContainsEmptyPostScript  = 0x0008
   };
   typedef long gxTranslationStatistic;
```

Macintosh Interface Functions

Associating View Ports With Macintosh Windows

```
gxViewPort GXNewWindowViewPort
                            (WindowPtr qdWindow);
gxWindowPtr GXGetViewPortWindow
                            (gxViewPort portOrder);
gxViewPort GXGetWindowViewPort
                            (WindowPtr qdWindow);
```

Associating View Devices With Macintosh Graphics Devices

```
GDHandle GXGetViewDeviceGDevice
                            (gxViewDevice theDevice);
gxViewDevice GXGetGDeviceViewDevice
                            (GDHandle qdGDevice);
```

Converting From QuickDraw to QuickDraw GX Coordinates

```
void GXConvertQDPoint       (const Point *shortPt, gxViewPort portOrder,
                             gxPoint *fixedPt);
void GXGetGlobalMouse       (gxPoint *globalPt);
void GXGetViewPortMouse     (gxViewPort portOrder, gxPoint *localPt);
```

Installing a View Port Filter

```
void GXSetViewPortFilter    (gxViewPort portOrder,
                             gxUserViewPortFilter filter, long refCon)
gxUserViewPortFilter GXGetViewPortFilter
                            (gxViewPort portOrder, long *refCon)
```

QuickDraw–to–QuickDraw GX Translator Functions

Converting QuickDraw Font and Style

```
long GXConvertQDFont        (gxStyle theStyle, long txFont, long txFace);
```

Converting QuickDraw Pictures

```
gxShape GXConvertPICTToShape
                            (const PicHandle pict,
                             gxTranslationOptions options,
                             const Rect *srcRect,
                             const Rect *dstRect,
                             Point styleStretch,
                             gxShape destination,
                             gxTranslationStatistics *stats);
```

Installing and Removing the Translator

```
void GXInstallQDTranslator  (GrafPtr port, gxTranslationOptions options,
                             const Rect *srcRect, const Rect *dstRect,
                             Point styleStretch,
                             gxShapeSpoolFunction userFunction,
                             long refCon);
translationStatistics *GXRemoveQDTranslator
                            (GrafPtr port,
                             translationStatistic *statistics);
```

Application-Defined Functions

Filtering Drawing Calls to a View Port

```
void MyViewPortFilter       (gxShape toFilter, gxViewPort portOrder,
                             long refCon);
```

Handling Translated QuickDraw Data

```
OSErr MyShapeSpooler        (gxShape toSpool, void *refCon);
```