

This chapter gives an overview of the typographic shape objects and the functions you can use to manipulate them. Read this chapter if you create or use any kind of QuickDraw GX typographic shapes.

Before reading this chapter, you should be familiar with the information in the chapter “Introduction to QuickDraw GX Typography” in this book. You also need to be familiar with the concept of objects as described in the book *Inside Macintosh: QuickDraw GX Objects*.

This chapter introduces the QuickDraw GX typographic shape object, explains how it relates to the concept of a QuickDraw GX shape object, and describes typographic shapes and their properties. It then shows how to use QuickDraw GX functions to

- hit-test typographic shapes
- measure typographic shapes
- convert typographic shapes to other types of shapes
- flatten typographic shapes
- use functions described in *Inside Macintosh: QuickDraw GX Objects* and *Inside Macintosh: QuickDraw GX Graphics* on typographic shapes

About Typographic Shapes

Typographic shapes are the QuickDraw GX shapes that display text: text shapes, glyph shapes, and layout shapes. In general, they act like other QuickDraw GX shapes, such as paths, polygons, or pictures. The geometry of the shape holds the unique characteristics of the typographic shape.

Each of the three shape types has different capabilities that make them suited for different uses. The needs of your application determine which type of shape you should use.

Types of Typographic Shapes

Text shapes contain the text you want to draw, one style to draw the text in, and a position at which to start drawing the text. They provide the simplest way for your application to draw text. Figure 2-1 shows how a text shape looks when drawn.

Figure 2-1 Text shape



The fifty bisected offices

Typographic Shapes

Note

In this book, the term *text* always refers to displayed writing. When referring to the text shape, this book always uses the term *text shape*. ♦

Glyph shapes allow you to draw text in several styles with independently positioned glyphs. You can give each glyph in the shape its own absolute position and draw each glyph at a different angle. In Figure 2-2, the glyph shape has two styles: most of the sentence is in regular Hoefler Text Italic, and the word “offices” is in bold. In addition, the shape is angled so that the first half of the sentence is at a 90-degree angle to the second half.

Figure 2-2 Glyph shape

**Note**

Do not confuse the term *glyph* with the shape type *glyph shape*, which is one of the typographic shapes. A glyph is a single unit of a font, and a glyph shape is a QuickDraw GX shape type that, when drawn, contains one or more glyphs. When referring to the shape, this book always uses the phrase *glyph shape*. ♦

Layout shapes display text in a typographically sophisticated manner. These shapes use more information from the font than other shapes do in order to display text correctly. Correct display might require kerning the text, drawing the glyphs in a left-to-right or right-to-left direction, or choosing different versions of glyphs depending on the glyph’s position in the text. For example, a layout shape containing Arabic text automatically draws the text from right to left and joins the glyphs together as required by the Arabic script. Figure 2-3 shows a layout shape with the same text as Figure 2-2. However, the layout shape uses information in the font to pick the best glyphs for display. Note that the uppercase “T” that begins the sentence is a variant form of the glyph that appears in Figure 2-2, because the “T” in Figure 2-3 is in a ligature. Also notice several other ligatures.

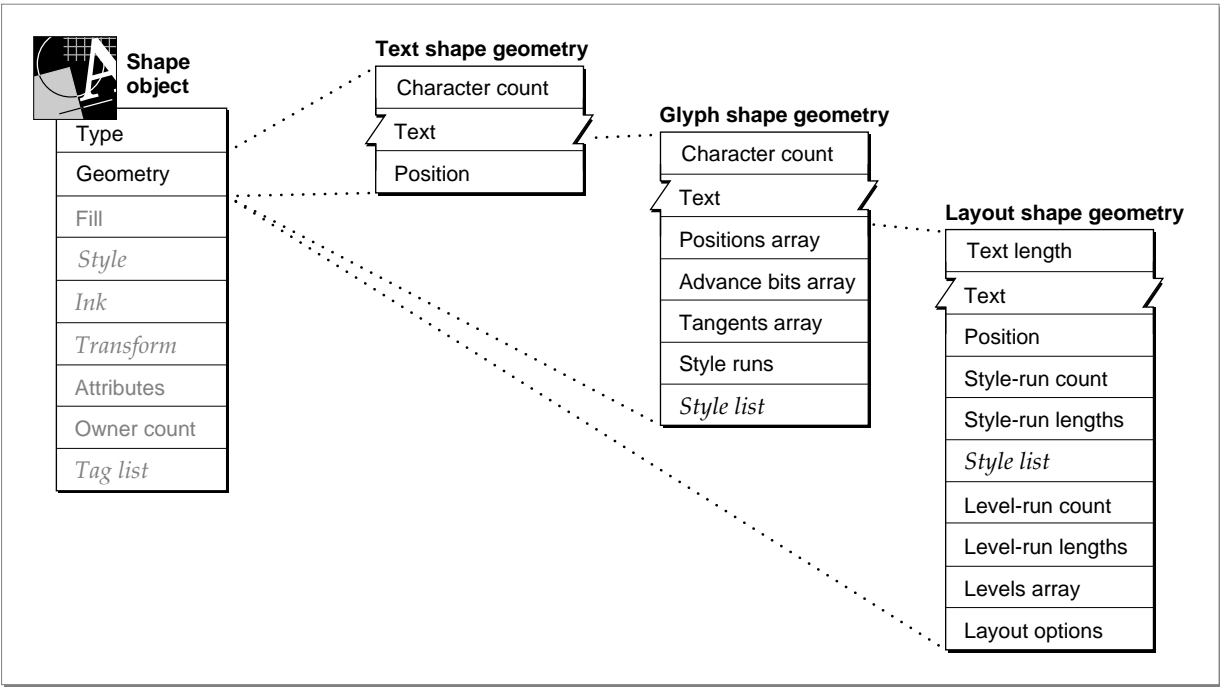
Figure 2-3 Layout shape



Typographic Shape Structure

Typographic shapes are organized like other shapes: they have the same shape object structure and have associated style, ink, and transform objects. Figure 2-4 shows the organization of a typographic shape, including the contents of its geometry.

Figure 2-4 Geometry of a typographic shape



Typographic Shapes

The geometry of a typographic shape contains the following elements in common:

- **Character count.** For text and glyph shapes only, the number of characters in the text array (byte count for layout shapes).
- **Text.** For glyph and layout shapes only, an array of character codes, glyph codes, or both character and glyph codes. For text shapes only, an array of character codes or glyph codes.
- **Position.** The starting position of the typographic shape in geometry space.

Note

The shape type property specifies whether the shape is a text, glyph, or layout shape. ♦

Typographic Shape Attributes

Typographic shapes, like other shapes, use shape attributes. The two shape attributes that apply most frequently to typographic shapes are the `gxMapTransformShape` and `gxIgnorePlatformShape` shape attributes.

- The `gxMapTransformShape` shape attribute applies transformation operations—for example, rotating, scaling, or skewing—to the shape’s transform object rather than changing the information in the shape object’s geometry. This attribute is set by default for layout shapes.
- The `gxIgnorePlatformShape` shape attribute indicates that QuickDraw GX should treat the codes in the geometry of this shape as glyph codes rather than as character codes. This attribute overrides information in the style object or style run arrays about the platform, script, and language used for individual style runs (explained in the chapter “Typographic Styles” in this book). The only time your application needs to set this attribute is if it needs to specify glyph codes directly—for example, if it is a font editor.

Default Characteristics of a Typographic Shape

The default settings for all typographic shapes are:

- **Geometry:** depends on the particular typographic shape. See the chapters “Text Shapes,” “Glyph Shapes,” and “Layout Shapes” in this book for specifics about the geometry of each of these shapes.
- **Fill:** winding fill. The valid fill types for typographic shapes are: winding, even-odd, and no fill.
- **Style:** the same default style object as other QuickDraw GX shapes. See the chapter “Typographic Styles” for more information about style properties, such as text attributes, that are specific to typographic shapes.
- **Ink:** the same default ink object as other shapes. Thus, the entire shape can only have one color, and all glyphs in the shape must be of the same color.
- **Transform:** the same default transform object as other shapes. The default layout shape has the `gxMapTransformShape` attribute set, whereas the other two typographic shapes, text and glyph shapes, do not. If the `gxMapTransformShape` attribute is set,

Typographic Shapes

any transformations that you apply to the shape are in fact applied to the mapping of the transform object. If the attribute is not set, the transformations are applied to the shape's geometry.

- Shape attributes: none. The exception is the layout shape, which has the `gxMapTransformShape` attribute set.
- Owner count: 1.
- Tags: none.

Typographic Shapes and the Style Object

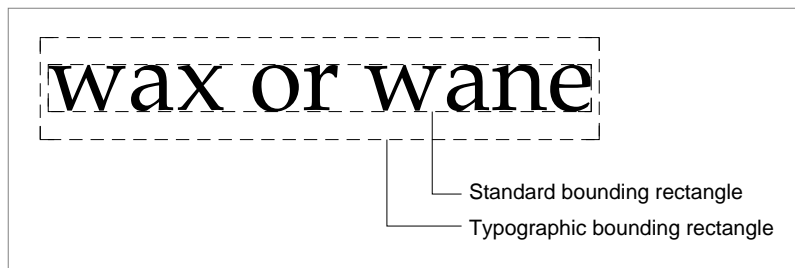
The style object associated with a typographic shape contains the font, the text size in points, and other information that determines the characteristics of a shape when it is displayed. (For more information about the parts of a style object that apply to typographic shapes, see the chapter “Typographic Styles” in this book.)

The Standard and Typographic Bounding Rectangles

Every QuickDraw GX shape has a standard bounding rectangle, which is the smallest rectangle that completely encloses the filled or framed parts of the shape. However, because of the height differences between glyphs—for example, between a small glyph, such as a lowercase “e”, and a taller, larger glyph, such as an uppercase “M”, or even between glyphs of different fonts and point sizes—the standard bounding rectangle may not be sufficient for your application's purposes. Therefore, you can also use the **typographic bounding rectangle**, which is the smallest rectangle that encloses the full span of the glyphs from the ascent line to the descent line.

Figure 2-5 shows an example of how the typographic bounding rectangle and standard bounding rectangle relate. The two rectangles are markedly different because the text has no ascenders or descenders. Whereas the standard bounding rectangle encloses just the black bits of the shape, the typographic bounding rectangle takes into account the ascent and descent lines for the shape. If the text in the figure includes glyphs with ascenders and descenders, the typographic bounding rectangle doesn't change, but the standard bounding rectangle does.

Figure 2-5 Standard bounding rectangle and typographic bounding rectangle



Using Typographic Shapes

This section describes the basic method of creating a typographic shape and some ways you might manipulate that shape. You can manipulate all three types of shapes in these ways, unless otherwise noted. For detailed information on using a specific typographic shape, look in this book for the chapter that describes that shape.

Because typographic shapes act like other types of shapes, you can use many of the functions described in *Inside Macintosh: QuickDraw GX Objects* on any of the typographic shapes. You can also use many of the functions described in *Inside Macintosh: QuickDraw GX Graphics* on the typographic shapes; these functions are mentioned in this section. This chapter assumes that you are either familiar with these functions already or have access to these other books—the function descriptions are not repeated here.

This section describes how you can

- position typographic shapes
- hit-test typographic shapes
- measure typographic shapes
- edit typographic shapes
- convert typographic shapes to other shape types
- insert part of a typographic shape into another shape
- flatten typographic shapes

Positioning Typographic Shapes

The initial position of your typographic shape is specified when the shape is created. In some cases, this will suffice. In other cases, you may wish to actually move the shape.

To move a shape to a specified position, use the `GXMoveShapeTo` function. This function positions the shape at a specified point and either changes the shape's transform object to reflect the move or changes the geometry of the shape, depending on whether the `gxMapTransformShape` shape attribute is set. For example, if you want to move the glyph origin of a typographic shape to the point (100.0,50.0), make this call:

```
GXMoveShapeTo(myGlyphShape, ff(100), ff(50));
```

The `GXMoveShapeTo` function can move a shape a specified distance rather than to a specified point, if the map transform bit is set and the shape has the identity transform.

You can also use the `GXMoveShape` function to move a shape a specified distance from its original position. This function moves the shape horizontally and vertically by the distances you specify. It either changes the shape's transform object to reflect the move or

Typographic Shapes

changes the geometry of the shape, depending on whether the `gxMapTransformShape` attribute is set. For example, if you want to move the glyph origin of a typographic shape down and to the right by 30 points, make this call

```
GXMoveShape(myTextShape, ff(30), ff(30));
```

The `GXMoveShapeTo` and `GXMoveShape` functions are described in *Inside Macintosh: QuickDraw GX Objects*.

Hit-Testing Typographic Shapes

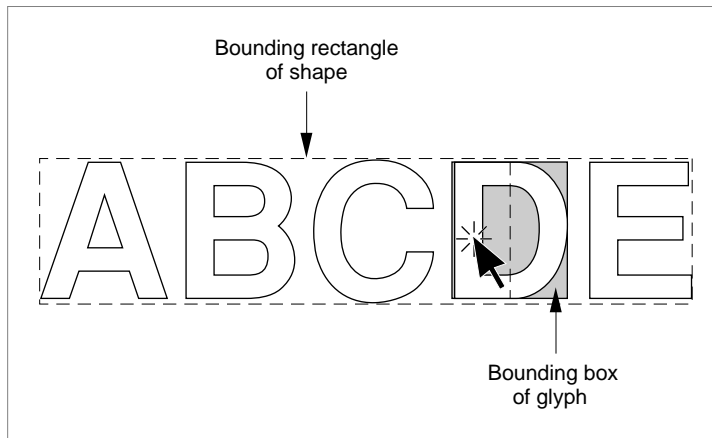
Hit-testing is a way of determining where the user clicked. To hit-test a typographic shape, QuickDraw GX provides two functions: `GXHitTestShape` and `GXHitTestLayout`.

Using GXHitTestShape

You can use the `GXHitTestShape` function to hit-test any kind of shape. This function takes parameters specifying the shape and the point where the user clicked, and it returns a `gxHitTestInfo` structure, which contains information about the specified point: its location in the shape and its distance from the bounding box of the glyph, depending on the hit-test parameters.

Figure 2-6 shows an example of hit-testing a typographic shape.

Figure 2-6 Hit-testing a typographic shape



The `GXHitTestShape` function and the `gxHitTestInfo` structure are described in *Inside Macintosh: QuickDraw GX Objects*. Functions specifically designed to hit-test layout shapes are described in the chapter “Layout Carets, Highlighting, and Hit-Testing” in this book.

Typographic Shapes

Using GXHitTestLayout

You can use the `GXHitTestLayout` function to convert a view port location (representing, for example, the position of a mouse-down event) into an edge offset in the source text of a layout shape. The `GXHitTestLayout` function determines which part of which glyph in the display text of a layout shape is closest to a particular location.

The `GXHitTestLayout` function is described in the chapter “Layout Carets, Highlighting, and Hit-Testing” in this book.

Measuring Typographic Shapes

You can measure typographic shapes exactly as you would any other type of shape: using, for example, the `GXGetShapeArea` function. This function returns the area covered by the shape’s standard bounding rectangle. You can also measure a typographic shape in different ways: you can get the bounding boxes of the individual glyphs in the shape or the typographic bounding rectangle of the shape.

This section describes how you can

- get the area of a typographic shape
- get and set the standard bounding rectangle
- get the font measurements from a typographic shape
- get the typographic bounding rectangle

Functions specific to the layout shape that measure line lengths and line spans are described in the chapter “Layout Line Control” in this book.

Getting the Area of a Typographic Shape

To get the area of the shape in square points, you can use the `GXGetShapeArea` function. For example, if you want the total area of a layout shape, send the function a reference to the shape, an index of 0, and a reference to a variable that will hold the result of the function:

```
GXGetShapeArea(myLayoutShape, 0, &theArea);
```

The `GXGetShapeArea` function converts a copy of the shape into a path before computing the area. The index can be 0, to refer to all paths, or an index to a specific contour. Because contours don’t necessarily have a one-to-one correspondence to characters or glyphs, it is rarely useful to specify a nonzero index for typographic shapes.

The function measures the shape area as defined in the shape’s geometry; it does not consider transformations to the shape made by the shape’s transform object.

The `GXGetShapeArea` function is described in *Inside Macintosh: QuickDraw GX Graphics*.

Getting and Setting the Standard Bounding Rectangle

To get the bounding rectangle of the shape, which is the smallest rectangle that completely encloses the filled part of the shape, you can use the `GXGetShapeBounds` function, which is described in *Inside Macintosh: QuickDraw GX Graphics*.

You can change the size of a typographic shape using the `GXSetShapeBounds` function, which is also described in *Inside Macintosh: QuickDraw GX Graphics*. By changing the bounding rectangle of the shape, you can change the width and height of the glyphs in the shape.

Figure 2-7 shows how decreasing the bounding rectangle can affect the size of the resulting shape. As with other shape types, the `gxMapTransformShape` attribute of the source shape determines how the function changes the bounding rectangle. If this attribute is set, the function does not alter the shape's geometry directly; if it is not, the function changes the geometry of the source shape to fit the new bounding rectangle.

Figure 2-7 Effects of the `GXSetShapeBounds` function



Getting the Font Measurements From a Typographic Shape

To get the glyph origins, bounding boxes, and left-side bearings of the glyphs in the typographic shape, you can use the `GXGetGlyphMetrics` function, which is described on page 2-24.

Getting the Typographic Bounding Rectangle

To get the typographic bounding rectangle of the shape, you can use the `GXGetShapeTypographicBounds` function, which is described on page 2-26. The function returns a rectangle that completely encloses the shape, from the highest ascent line to the lowest descent line in the shape. (Ascent lines and descent lines are described in the chapter “Introduction to QuickDraw GX Typography” in this book.

You cannot set the typographic bounding rectangle, because the measurements of the glyphs in the shape are determined by the font, not by QuickDraw GX.

Editing Typographic Shapes

In general, you should use the `GXSetShapeType` and `GXSetShapeTypeParts` functions to edit typographic shapes. Each shape—text, glyph, or layout—has its own specific function, such as `GXSetTextParts`. These functions are more efficient than `GXSetShapeType` functions because you don't have to replace all of the information in a shape at once, as you do with the `GXSetShapeType` functions. Instead, you can replace specific parts of the shape, such as inserting text into the shape's text array, quickly. For example, use `GXSetLayoutParts` to append a single character to an existing layout shape.

Converting Typographic Shapes

You can convert one typographic shape to its primitive form or to a geometric shape, bitmap, or picture. This section describes these operations.

Converting a Typographic Shape to Its Primitive Form

You can use the `GXPrimitiveShape` function, described in *Inside Macintosh: QuickDraw GX Graphics*, to convert any of the typographic shapes to their primitive forms. The primitive form of a typographic shape is a glyph shape, unless the typographic shape has one or more text faces, in which case its primitive form is a path shape.

The primitive glyph shape contains simple styles in its style run; none of the style run entries are `nil`, and the styles do not contain tags, caps, dashes, patterns, joins, font variations, text faces, or any of the layout style features, such as run controls, justification overrides, glyph substitutions, run features, or kerning adjustments. (Font variations and text faces are discussed in the chapter “Typographic Styles” in this book. Caps, dashes, patterns, and joins are discussed in the chapter “Geometric Styles” in *Inside Macintosh: QuickDraw GX Graphics*.)

The `GXPrimitiveShape` function replaces any `nil` styles in the style list with a reference to the shape's style object.

Converting Typographic Shapes to Other Shape Types

You can change any type of typographic shape to a geometric shape, bitmap shape, picture shape, an empty shape, or a full shape, with the varying results shown in Table 2-1.

You cannot, however, change other types of shapes to typographic shapes. If you try to convert the data from a geometric shape, a bitmap shape, or a picture shape to a typographic shape, QuickDraw GX always posts the error `new_shape_contains_invalid_data`.

Table 2-1 Results of converting typographic shapes to other types of shapes

New shape type	Result
Point	A point equal to the upper-left corner of the bounding rectangle of the original shape.
Line	A line from the upper-left corner to the lower-right corner of the bounding rectangle of the original shape.
Curve	A point equal to the glyph origin for the original shape.
Rectangle	The bounding rectangle of the original shape.
Polygon	A polygon of the text from the original shape.
Path	A path shape that traces the text from the original shape.
Bitmap	A bitmap of the text from the original shape.
Picture	A picture of the text from the original shape.
Empty	An empty shape.
Full	A full shape.

You can change any type of typographic shape into another typographic shape with the varying results shown in Table 2-2.

Table 2-2 Converting a typographic shape to another typographic shape

Source shape	Target shape	Result
Glyph shape	Text shape	The source text is gathered together and set as the text shape's text, but the new text shape gets the default style (which may not match the style information of the original glyph shape). None of the tangent, positioning, or advance bit information is preserved (except for the initial position, which is set as the position of the text shape).
Glyph shape	Layout shape	The source text and style run information is converted. Any specified tangents, advance bits, or positions are not included in the resulting layout shape (except for the initial position, which is set as the position of the layout shape).
Text shape	Glyph shape	A one-run glyph shape using the text shape's text and the default style is the result.
Text shape	Layout shape	A one-run layout using the text shape's text and the default style is the result.

continued

Typographic Shapes

Table 2-2 Converting a typographic shape to another typographic shape (continued)

Source shape	Target shape	Result
Layout shape	Glyph shape	This conversion can be done in one of two ways. If <code>GXSetShapeType</code> is used, the resulting glyph shape has the same geometry as the layout shape, with none of the layout effects. If <code>GXPrimitiveShape</code> is used, the resulting glyph shape has all the layout effects.
Layout shape	Text shape	The source text is gathered together and set as the text shape's text, but the new text shape gets the default style (which may not match the style information of the original glyph shape). None of the layout effects are preserved.

Inserting Part of a Typographic Shape Into Another Shape

You can use the `GXGetShapeParts` and `GXSetShapeParts` functions, described in the chapter “Geometric Shapes” in *Inside Macintosh: QuickDraw GX Graphics*, to insert parts of one typographic shape into another typographic shape. You can also use the `GXSetShapeTypeParts` functions described in the chapters “Text Shapes,” “Glyph Shapes,” and “Layout Shapes” in this book.

Note

When you use these functions with typographic shapes, a “shape part” consists of one or more characters or glyphs and any styles that apply to those characters or glyphs. These functions do not use the hit-testing `gxShapeParts` enumeration, described on page 2-23. ♦

For example, suppose a text shape contains “abcd” and a glyph shape contains “efgh”. The following code fragment inserts the first three glyphs from the text shape (“abc”) into the middle of the glyph shape (after “f”):

```
GXGetShapeParts(myTextShape, 1, 3, myInsertShape);
GXSetShapeParts(myGlyphsShape, 2, 0, myInsertShape,
myEditShapeFlags);
```

The result of this code would be a glyph shape that reads “efabcgh”.

If you want to put part of a geometric shape, a full shape, or a picture shape into a glyph or layout shape, the `GXSetShapeParts` function converts the part to be inserted to the empty shape. If you want to insert the part into a text shape, the function converts both shapes to path shapes. Table 2-3 lists the behavior for most of the general operations involving the `GXSetShapeParts` function.

Table 2-3 Setting the shape parts of various types of shapes

Source shape	Target shape	Action
Typographic	Point, line	Changes a typographic shape to a polygon shape.
Typographic	Curve, path	Changes a typographic shape to a path shape.
Typographic	Rectangle	Posts the error <code>rectangle_cannot_be_inserted_into</code> .
Typographic	Bitmap	Posts the warning <code>shape_operator_may_not_be_a_bitmap</code> .
Typographic	Picture	Replaces the specified shape or shapes in the picture with the parts of the source shape.
Glyph, layout	Text	Changes a text shape to a glyph or layout shape.
Geometry, full, picture	Glyph, layout	Converts the source shape to an empty shape and posts the warning <code>shape_does_not_contain_text</code> .
Bitmap	Glyph, layout	Posts the warning <code>shape_operator_may_not_be_a_bitmap</code> .
Geometry, full, picture	Text	Changes both shapes to path shapes.

Note

The layout shape has its own functions for getting and setting shape parts. The functions `GXSetLayoutShapeParts` and `GXGetLayoutShapeParts` are described in the chapter “Layout Shapes” in this book. The `GXGetTextParts` and the `GXSetTextParts` functions are described in the chapter “Text Shapes” in this book. The `GXGetGlyphParts` and the `GXSetGlyphParts` functions are described in the chapter “Glyph Shapes” in this book. ♦

Flattening Typographic Shapes

You can use the `GXFlattenShape` function (described in *Inside Macintosh: QuickDraw GX Objects*) on typographic shapes exactly as you would on any other type of shape. Flattening a shape, however, does not flatten the fonts that are in that shape. It does flatten all the style objects in the style list that are part of the geometry of a glyph or layout shape.

When you flatten a shape that contains fonts, QuickDraw GX creates a **flat font list**. This list specifies which fonts were used in the shape, which glyphs in that font were used in the shape, or both. Each entry in the flat font list has the tag `'flst'`.

To flatten the fonts in the shape and include them along with the flattened shape, you can use the entries in the flat font list.

Typographic Shapes

For more information about flattening shapes, see the chapter “Shape Objects” of *Inside Macintosh: QuickDraw GX Objects* and the chapter “QuickDraw GX Stream Format” of *Inside Macintosh: QuickDraw GX Environment and Utilities*.

Applying Functions Described Elsewhere to Typographic Shapes

QuickDraw GX provides only a small number of functions that apply exclusively to typographic shapes. However, most of the QuickDraw GX functions that apply to other types of shapes can also be applied to typographic shapes.

The next four sections give an overview of these functions and their effects on typographic shapes.

- “Shape-Related Functions,” lists functions that operate on typographic shape objects and geometric operations that work for typographic shape geometries.
- “Style-Related Functions” on page 2-20 discusses how style-related functions affect the drawing of typographic shapes.
- “Ink- and Color-Related Functions” on page 2-20 discusses how the transfer mode of a typographic shape’s ink object is used to draw a typographic shape.
- “Transform- and View-Related Functions” on page 2-20 lists the functions that allow you to map and clip a typographic shape as well as set its hit-test parameters and its view-port list. This section also includes the functions that manipulate the typographic shape associated with a view device object.

Shape-Related Functions

You can apply all of the functions described in the “Shape Objects” chapter of *Inside Macintosh: QuickDraw GX Objects* to typographic shapes. These functions allow you to:

- manipulate the shape object that represents the typographic shape; for example, copy, clone, cache, compare, and dispose of the typographic shape
- set the geometry, shape type, shape fill, and shape attributes of the typographic shape
- change the style, ink, and transform objects that are associated with the typographic shape
- manipulate the typographic shape’s tags and owner count

Table 2-4 gives a partial list of the functions from the chapter “Shape Objects” in *Inside Macintosh: QuickDraw GX Objects*. You should be aware of the results these functions have when you apply them to typographic shapes. All other general shape functions should act on typographic shapes exactly as they do on any other type of shape.

Typographic Shapes

Table 2-4 Selected effects of shape-related functions that you can apply to typographic shapes

Function name	Action taken
GXCopyToShape	Makes a copy of the typographic shape. It does not copy the styles in the styles list (for glyph and layout shapes).
GXCopyDeepToShape	Makes a copy of the typographic shape, including copies of the style in the styles list (for glyph and layout shapes).
GXGetShapeStyle	Returns the style object associated with the typographic shape.

You can also apply certain geometric shape functions to typographic shapes. Table 2-5 gives a selected list of the functions from the chapter “Geometric Shapes” in *Inside Macintosh: QuickDraw GX Graphics*. You should be aware of the effects of these functions have when you apply them to typographic shapes.

Table 2-5 Geometric shape functions that you can apply to typographic shapes

Function name	Action taken
GXCountShapeContours	For text and layout shapes, the function returns the number of bytes. For glyph shapes, it returns the number of characters.
GXCountShapePoints	Returns 1 for text shapes. For glyph and layout shapes, the <code>contour</code> parameter specifies the style run about which you want information (or, if you pass 0, the entire shape). For glyph shapes, the function returns the number of characters in the specified style run or in the shape. For layout shapes, the function returns the number of bytes in the specified style run or in the shape.
GXGetShapeIndex	Posts the warning <code>graphic_type_does_not_have_multiple_contours</code> and returns 0.
GXGetShapePoints	For text and layout shapes, the function always returns 1. For glyph shapes, it returns the positions array of the shape and the number of entries in the array.
GXSetShapePoints	For text and layout shapes, sets the initial glyph position of the shape. For glyph shapes, the function sets elements in the positions array.

You can also apply certain geometric operation functions to typographic shapes. Table 2-6 gives partial list of the functions from the chapter “Geometric Operations” in *Inside Macintosh: QuickDraw GX Graphics*. You should be aware of the effects these functions have when you apply them to typographic shapes. All other functions in this chapter should act on, or have no meaning for, typographic shapes.

Typographic Shapes

Table 2-6 Geometric operations that you can apply to typographic shapes

Function name	Action taken
<code>GXBreakShape</code>	Converts text and layout shapes into glyph shapes. On glyph shapes, the function uses the <code>index</code> parameter as a character index and splits the style run at that index into two runs.
<code>GXContainsBoundsShape</code>	Returns <code>true</code> if the rectangle indicated in the <code>container</code> parameter contains the bounding rectangle of the untransformed typographic shape; returns <code>false</code> otherwise.
<code>GXContainsShape</code>	Treats both parameters of the function as though they were path shapes. The function returns <code>true</code> if the <code>container</code> parameter is equal to or larger than the area of the <code>test</code> parameter.
<code>GXGetShapeDirection</code>	Posts the error <code>illegal_type_for_shape</code> .
<code>GXReverseShape</code>	Posts the warning <code>contour_out_of_range</code> .
<code>GXReduceShape</code>	Posts the warning <code>graphic_type_cannot_be_reduced</code> .
<code>GXSimplifyShape</code>	Posts the notice <code>shape_already_in_simple_form</code> .
<code>GXPrimitiveShape</code>	Creates the primitive form of the typographic shape.
<code>GXGetShapeLength</code>	Posts the warning <code>shape_does_not_have_length</code> .
<code>GXShapeLengthToPoint</code>	For text and glyph shapes, posts the warning <code>shape_does_not_have_length</code> .
<code>GXGetShapeArea</code>	Converts layout shapes to glyph shapes before continuing. For text and glyph shapes, the function returns the weighted center of all of the bounding boxes of all glyphs (excluding glyphs without contours, such as the space glyph).
<code>GXGetShapeCenter</code>	Returns the point that is the center of the shape's standard bounding rectangle.
<code>GXGetShapeBounds</code>	Returns the bounds of the black bits area of the specified glyph in the shape or the bounds of all the glyphs in the shape if you specify an index of 0. For layout shapes, the function returns the bounds as if the layout shape had been converted to a glyph shape by <code>GXPrimitiveShape</code> .

continued

Table 2-6 Geometric operations that you can apply to typographic shapes (continued)

Function name	Action taken
GXSetShapeBounds	If the new bounds is a translation of the old bounds or if the scaling is square, changes the shape type to a glyph shape and changes the tangents array of the shape. Keep in mind that if the <code>gxMapTransformShape</code> attribute is set, this function changes the typographic shape's transform mapping; otherwise, it changes the geometry. If the scaling is not square, the function changes the shape to a path shape.
GXInsetShape	Moves the on-curve control points of each glyph by the amount you specify. If you specify very small values for the <code>inset</code> parameter, you can use the function to make the glyphs of the shape thicker or thinner, creating a bold or thin look for the glyphs. However, large inset values are discouraged, because they may distort the glyphs beyond recognition.
GXTouchesBoundsShape	Treats the typographic shape as if it were a path shape, although the function does not change the shape's actual type. The function returns <code>true</code> if the rectangle indicated in the <code>target</code> parameter intersects the bounding rectangle of the path shape described by the untransformed typographic shape; returns <code>false</code> otherwise.
GXTouchesShape	Returns <code>true</code> if two shapes intersect. One or both of the shapes may be typographic shapes.
GXInvertShape	Posts the warning <code>shape_cannot_be_inverted</code> .
GXIntersectShape	If the typographic shape is the shape specified by the <code>target</code> parameter, these functions convert the shape to a path shape in most cases. Otherwise, these functions act exactly as they do for other shape types.
GXUnionShape	
GXDifferenceShape	
GXReverseDifferenceShape	
GXExcludeShape	

The other functions described in the chapter “Geometric Operations” are primarily designed for the geometric, bitmap, and picture shape types. These functions and the errors and warnings that they post are described in *Inside Macintosh: QuickDraw GX Graphics*.

Style-Related Functions

Functions and attributes related to how typographic shapes use the style object are described in the chapter “Typographic Styles” in this book.

Although you can use certain functions described in *Inside Macintosh: QuickDraw GX Graphics* (such as `GXSetShapePen`, `GXSetShapeDash`, and so on) to set the other properties of a typographic shape’s style object and other corresponding functions (`GXGetShapePen`, `GXGetShapeDash`, and so on) to examine those properties, QuickDraw GX ignores these properties when drawing a typographic shape. It does use them, however, when these properties are part of a text face’s style. (See the chapter “Typographic Styles” for more information on how the style object can use geometric properties.)

Ink- and Color-Related Functions

You can use only a single ink object when drawing a typographic shape; thus, the entire shape must share the same color. QuickDraw GX considers the transfer mode of the ink object and applies it when drawing a typographic shape.

Transform- and View-Related Functions

You can apply all of the transform-related functions to typographic shapes. If the `gxMapTransformShape` shape attribute is set, all transformations are applied to the mapping matrix in the shape’s transform object. If the attribute is not set, the transformations are applied to the shape’s geometry. (The layout shape, by default, has the `gxMapTransformShape` shape attribute set; the text and glyph shapes do not.)

Table 2-7 gives a partial list of the functions from the “Transform Objects” chapter of *Inside Macintosh: QuickDraw GX Objects*. You should be aware of the effects these functions have when you apply them to typographic shapes. All other functions should act on typographic shapes exactly as they do on any other type of shape.

Typographic Shapes Reference

This section describes constants, data types, and functions that are useful for all three typographic shapes: text shapes, glyph shapes, and layout shapes.

Each shape type also has its own constants, data types, and functions, in addition to the ones described in this chapter. See the chapters “Text Shapes,” “Glyph Shapes,” and “Layout Shapes” in this book.

Table 2-7 Selected transform-related functions that you can apply to typographic shapes

Function name	Action taken
GXMapShape	Changes the mapping associated with the typographic shape's transform (if the <code>gxMapTransformShape</code> attribute of the typographic shape is set) or applies the mapping directly to the geometry of the typographic shape (if the <code>gxMapTransformShape</code> attribute is not set).
GXMoveShape	Changes the typographic shape position by specified offsets or applies the mapping directly to the transform of the typographic shape (if the <code>gxMapTransformShape</code> attribute is set). If the shape is a glyph shape, the function changes the positions arrays accordingly.
GXMoveShapeTo	Sets the typographic shape position or applies the mapping directly to the transform of the typographic shape (if the <code>gxMapTransformShape</code> attribute is set). If the shape is a glyph shape, the function changes the positions arrays accordingly.
GXRotateShape	Rotates the typographic shape. This function can affect the mapping of the typographic shape's transform if the <code>gxMapTransformShape</code> attribute is set. Otherwise, it converts a text shape to a glyph shape, and it changes the values in the glyph shape's tangents array. You use the <code>GXRotateShape</code> function to create vertical text. See the chapter "Typographic Styles" for more information on vertical text.
GXScaleShape	Scales the typographic shape. If the shape's <code>gxMapTransformShape</code> attribute is not set, the function changes the shape to a glyph shape. (Otherwise, the function changes the shape's transform.) If you specify square scaling, the function changes the tangents and positions arrays of the glyph shape. Otherwise, the function changes the shape to a path shape.
GXSkewShape	Skews the glyph shape. If the shape's <code>gxMapTransformShape</code> attribute is not set, the function changes the shape to a path shape. (Otherwise, the function changes the shape's transform.)

Constants and Data Types

This section describes how those parts of the `gxShapeAttributes` enumeration and the `gxShapeParts` enumeration that apply specifically to typographic shapes. The `gxShapeAttributes` enumeration is described in more detail in the "Shape Objects" chapter in *Inside Macintosh: QuickDraw GX Objects*; the `gxShapeParts` enumeration is described in the "Transform Objects" chapter of *Inside Macintosh: QuickDraw GX Objects*.

Shape Attributes

Each shape object has a set of shape attributes. **Shape attributes** are a group of flags that modify the behavior of the shape object. The shape attributes are defined as follows:

```
enum gxShapeAttributes{
    gxNoAttributes,
    gxDirectShape          = 0x0001,
    gxRemoteShape          = 0x0002,
    gxCachedShape          = 0x0004,
    gxLockedShape          = 0x0008,
    gxGroupShape           = 0x0010,
    gxMapTransformShape     = 0x0020,
    gxUniqueItemsShape      = 0x0040,
    gxIgnorePlatformShape   = 0x0080,
    gxNoMetricsGridShape    = 0x0100,
    gxDiskShape            = 0x0200,
    gxMemoryShape           = 0x0400
};
```

```
typedef long gxShapeAttribute;
```

The following constants are of particular use for typographic shapes:

Constant descriptions

gxMapTransformShape

Indicates that any transforms or mappings you apply to the shape are applied to the shape's transform object. This bit is set by default for layout shapes. If this bit is not set, transforms and mappings are applied to the shape's geometry.

gxIgnorePlatformShape

Indicates that the typographic shape contains only glyph codes, which are 16-bit long values. This value overrides the current setting of the platform value of the style object. Setting this attribute is equivalent to setting the platform of all the styles attached to a shape to `gxGlyphPlatform`.

gxNoMetricsGridShape

Indicates that QuickDraw GX is not to use hints that may be provided by a font. Set this attribute if you intend to manipulate text as a shape. The hinting can affect a shape's geometry, which may be undesirable if you want to perform other operations, such as scaling, on the shape.

All other values in the `gxShapeAttributes` enumeration are described in the chapter "Shape Objects" in *Inside Macintosh: QuickDraw GX Objects*. Font platforms are described in the chapter "Font Objects" in this book.

Shape Parts

To determine exactly where the user clicked on a glyph, use the values of the `gxShapeParts` enumeration.

```
enum gxShapeParts { /* parts of a gxShape in hit-testing: */
    gxNoPart          = 0,          /* (in order of evaluation) */
    gxBoundsPart      = 0x0001,
    gxGeometryPart    = 0x0002,
    gxPenPart         = 0x0004,
    gxCornerPointPart = 0x0008,
    gxControlPointPart = 0x0010,
    gxEdgePart        = 0x0020,
    gxJoinPart        = 0x0040,
    gxStartCapPart    = 0x0080,
    gxEndCapPart      = 0x0100,
    gxDashPart        = 0x0200,
    gxPatternPart     = 0x0400,
    gxGlyphBoundsPart = gxJoinPart,
    gxGlyphFirstPart  = gxStartCapPart,
    gxGlyphLastPart   = gxEndCapPart,
    gxSideBearingPart = gxDashPart,
    gxAnyPart         = gxBoundsPart | gxGeometryPart |
        gxPenPart | gxCornerPointPart | gxControlPointPart |
        gxEdgePart | gxJoinPart | gxStartCapPart | gxEndCapPart |
        gxDashPart | gxPatternPart
};

typedef long gxShapePart;
```

The following constants are of particular use for typographic shapes.

Constant type descriptions

<code>gxBoundsPart</code>	Indicates that the user clicked within the bounds rectangle that surrounds the shape. You can get this rectangle using the <code>GXGetShapeBounds</code> function, described in <i>Inside Macintosh: QuickDraw GX Graphics</i> .
<code>gxCornerPointPart</code>	Indicates that the user clicked on the glyph advance starting pen position.
<code>gxControlPointPart</code>	Indicates that the user clicked on the character advance ending pen position.
<code>gxEdgePart</code>	Indicates that the user clicked on the line defined by the advance vector (starting pen position to ending pen position).

Typographic Shapes

`gxGlyphBoundsPart`

Indicates that the user clicked in the bounding box of the glyph.

`gxGlyphFirstPart`

Indicates that the user clicked on the left or top side of the glyph (depending on the rotation of the shape).

`gxGlyphLastPart` Indicates that the user clicked on the right or bottom side of the glyph (depending on the rotation of the shape).

`gxSideBearingPart`

Indicates that the user clicked in the side bearing of the glyph. You can use this value in combination with either the `gxGlyphFirstPart` or `gxGlyphLastPart` values to determine whether the user clicked on the left or right (top or bottom) side of the glyph.

All other values in the `gxShapeParts` enumeration are described in the “Transform Objects” chapter of *Inside Macintosh: QuickDraw GX Objects*.

Functions

This section describes the functions that allow you to get the measurements of the advance widths, bounding boxes, and side bearings of any of the typographic shapes.

Measuring Typographic Shapes

The `GXGetGlyphMetrics` function returns the metrics of the glyphs produced by the shape. The measurements for these glyphs may change depending on the font, text size, platform value, text direction (horizontal or vertical), and other variables.

The `GXGetShapeTypographicBounds` function returns the typographic bounding rectangle. (See the section “The Standard and Typographic Bounding Rectangles” on page 2-7 for more information about the typographic bounding rectangle.)

GXGetGlyphMetrics

You can use the `GXGetGlyphMetrics` function to determine the metrics of the glyphs in any typographic shape.

```
long GXGetGlyphMetrics(gxShape source, gxPoint glyphOrigins[],
                       gxRectangle boundingBoxes[],
                       gxPoint sideBearings[]);
```

`source` A reference to the typographic shape (text, glyph, or layout) whose metrics you want to determine.

Typographic Shapes

`glyphOrigins`

An array of point structures. On return, the array contains the glyph origins, as points in the view port, for the glyphs in the shape. The array always contains one entry *more* than the number of glyphs in the shape. The last entry is the position of the end of the advance width of the final glyph in the shape. This array is optional; you may pass `nil` for this parameter.

`boundingBoxes`

An array of rectangle structures. On return, the array specifies the bounding boxes for the black portion of each glyph; there is one entry in the array for each glyph in the shape. The bounding boxes are relative to the origin of the shape, not to the beginning of the glyph that the bounding box describes. This array is optional; you may pass `nil` for this parameter.

`sideBearings`

An array of point structures. On return, the array contains the vectors along the advance between the pen position and the glyph's bounding box; there is one entry in the array for each glyph in the shape. This array is optional; you may pass `nil` for this parameter.

function result The number of glyphs in the shape.

DESCRIPTION

The `GXGetGlyphMetrics` function returns the number of glyphs in the shape. The function also returns the various metrics that describe the text, glyph, and layout shapes. Note that the glyph metrics returned by the `GXGetGlyphMetrics` function are device metrics, which are specific to the device on which you are rendering the shape, and not ideal metrics, which are device-independent.

The glyph origins array has one more entry than the number of glyphs in the shape. Be aware that the first entry in the array may not correspond to the given starting position. For example, suppose the layout shape “office” has its position set to (100.0,100.0) and uses hanging punctuation. The entry in the advance bits array for the open quotation mark could be (92.0,100.0) and the entry for the “o” could be (99.0,100.0), because of the optical alignment of the glyph. (Hanging glyphs and optical alignment are described in the chapter “Layout Styles” in this book.)

The final position in the glyph origins array is the position of the end of the advance width of the final glyph. For example, if the origin of the final glyph is (50.0,50.0), and the glyph is 10 points wide, the last entry in the array will be (60.0,50.0).

ERRORS, WARNINGS, AND NOTICES

Errors

`illegal_type_for_shape` (if not typographic) (debugging version)
`shape_is_nil`

You can use the `GXGetShapeTypographicBounds` function to get the typographic bounding rectangle of any typographic shape.

source	A reference to the shape whose typographic bounding rectangle you want.
rect	A pointer to a rectangle structure. On return, the structure contains the typographic bounding rectangle of the source shape.

DESCRIPTION

For shape types other than typographic shapes, the function posts an error.

Errors

Typographic Shapes Reference

Summary of Typographic Shapes

Constants and Data Types

Shape Attributes

```
enum gxShapeAttributes{
    gxNoAttributes,
    gxDirectShape          = 0x0001,
    gxRemoteShape          = 0x0002,
    gxCachedShape          = 0x0004,
    gxLockedShape          = 0x0008,
    gxGroupShape           = 0x0010,
    gxMapTransformShape    = 0x0020,
    gxUniqueItemsShape     = 0x0040,
    gxIgnorePlatformShape  = 0x0080,
    gxNoMetricsGridShape   = 0x0100,
    gxDiskShape            = 0x0200,
    gxMemoryShape          = 0x0400
};
```

```
typedef long gxShapeAttribute;
```

Shape Parts

```
enum gxShapeParts { /* parts of a gxShape in hit-testing: */
    gxNoPart              = 0,      /* (in order of evaluation) */
    gxBoundsPart          = 0x0001,
    gxGeometryPart        = 0x0002,
    gxPenPart             = 0x0004,
    gxCornerPointPart     = 0x0008,
    gxControlPointPart    = 0x0010,
    gxEdgePart            = 0x0020,
    gxJoinPart            = 0x0040,
    gxStartCapPart        = 0x0080,
    gxEndCapPart          = 0x0100,
    gxDashPart            = 0x0200,
    gxPatternPart         = 0x0400,
    gxGlyphBoundsPart     = gxJoinPart,
    gxGlyphFirstPart      = gxStartCapPart,
```

Typographic Shapes

```

gxGlyphLastPart      = gxEndCapPart,
gxSideBearingPart    = gxDashPart,
gxAnyPart             = gxBoundsPart | gxGeometryPart |
    gxPenPart | gxCornerPointPart | gxControlPointPart |
    gxEdgePart | gxJoinPart | gxStartCapPart | gxEndCapPart |
    gxDashPart | gxPatternPart
};

typedef long gxShapePart;

```

Functions

Measuring Typographic Shapes

```

long GXGetGlyphMetrics      (gxShape source, gxPoint glyphOrigins[],
                             gxRectangle boundingBoxes[],
                             gxPoint sideBearings[]);

gxRectangle *GXGetShapeTypographicBounds
    (gxShape source, gxRectangle *rect);

```