

Text Shapes

This chapter defines the QuickDraw GX text shape, describes what it contains, and tells how you can create one. Read this chapter if your application has simple text requirements, such as displaying text in a single font, text size, and typestyle.

Before reading this chapter, you should be familiar with the information in *Inside Macintosh: QuickDraw GX Objects*. You should also be familiar with the information in the chapters “Introduction to QuickDraw GX Typography” and “Typographic Shapes” in this book.

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

- create and draw text shapes
- change parts of text shapes

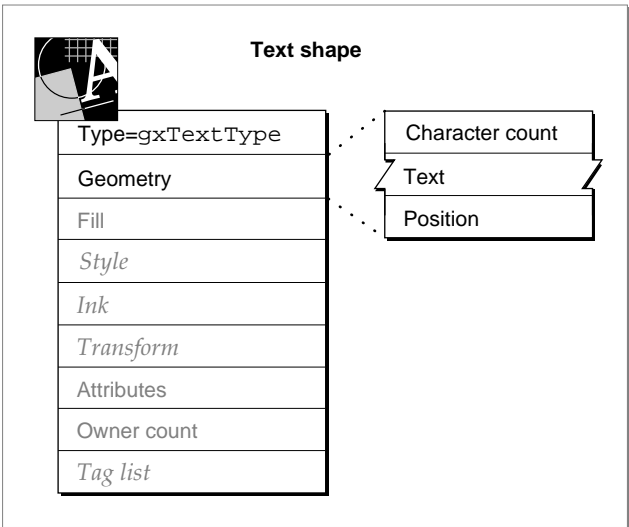
About Text Shapes

A **text shape** is a shape object containing a string of text associated with a single style object. The text shape has the same set of properties as other QuickDraw GX shape objects. Its shape type is `gxTextType`, and its geometry is unique.

The Geometry of a Text Shape

Figure 3-1 shows the properties of the text shape object. Note that, because a text shape is an object and not a data structure, the order of the properties as shown in Figure 3-1 is completely arbitrary.

Figure 3-1 Geometry of a text shape



Text Shapes

The geometry of a text shape contains these elements:

- **Character count.** The number of characters in the text array. This number is not necessarily the same as the number of bytes in the text array, because some of the characters may be 16-bit characters.
- **Text.** An array of character codes or glyph codes. Whether each character code is 8-bit or 16-bit depends on both the platform in the text shape's style object and on the actual byte values. If the value of the shape attribute `gxIgnorePlatformShape` is clear, then the text is interpreted as an array of character codes. If it is set, it is interpreted as an array of 16-bit glyph codes. (For more information about platforms, character codes, and glyph codes, see the discussion of encodings in the chapter "Font Objects" in this book.)
- **Position.** A point that marks the glyph origin of the first glyph in the shape.

You use the functions described in this chapter to set the values of the character count, the text, and the position of the text shape.

Because a text shape is a single run, you cannot mix character codes and glyph codes in a text shape; one run can have either character codes or glyph codes—but not both. A text shape always contains one glyph per character, and they are always in the same order. In other words, the character-code index is equal to the glyph index.

Note

The text shape displays only the glyphs the user asks for. For example, if the user types the glyphs "f" and "i", the text shape does not automatically display an "fi" ligature. If you need automatic linguistic processing of glyphs, you should use the layout shape, described in the chapter "Layout Shapes" in this book. ♦

The Default Text Shape

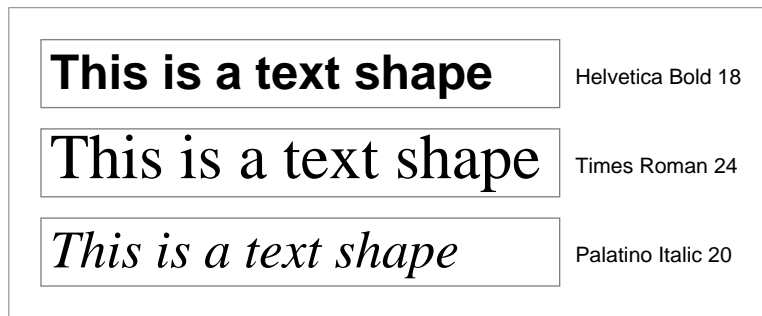
The default text shape has no text; all of its components have the value 0 or nil. Like the default glyph and layout shapes, the default text shape has the `gxWindingFill` type. The `gxWindingFill` type and other shape fills are described in the "Shape Objects" chapter of *Inside Macintosh: QuickDraw GX Objects*.

The default text shape has a style object, which is described in detail in the "Typographic Styles" chapter in this book. The default settings for all typographic shapes are described in detail in the chapter "Typographic Shapes" in this book.

The Text Shape and Styles

The style object associated with a text shape contains the font, the text size in points, and other information that determines characteristics of the 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.) Because a text shape can have only one style object associated with it, the text of a text shape is displayed in a single style.

Figure 3-2 shows three different examples of a text shape, each with a different style applied to it.

Figure 3-2 Three examples of a text shape, each with a different style applied

Because the text shape has only one style, it is most useful for drawing nonformatted glyphs, such as those used in dialog boxes, terminal emulation programs, or primitive text editors. Because the text shape contains only basic data, it is more efficient where speed is concerned, but it contains less information than the other typographic shapes.

For more complex editors or word processors and for contextual non-Roman text, you should use the layout shape, described in the chapter “Layout Shapes” in this book.

Also, you cannot use a text shape for clipping, dashing, patterns, joins, and start and end caps. Before you can use these features, you must convert the text shape to a glyph shape. You can, however, pattern a text shape by including, in the text shape’s style object, a patterned text face. A text face can also contain joins, start and end caps, dashing, and so on. Text faces are described in the chapter “Typographic Styles” in this book.

In general, you shouldn’t use text shapes and glyph shapes for non-Roman text. For example, if you need to draw text vertically or text from right to left, you should use the layout shape. For more information, see the chapter “Layout Shapes” in this book.

Using Text Shapes

This section describes the functions you use to create and draw text shapes and to change the information in a text shape.

Creating and Drawing a Text Shape

To create a text shape, you can use the `GXNewShape` function, specifying `gxTextType`, and then set properties. Alternatively, you can use the `GXNewText` function, passing it the necessary information. For example, when you create a text shape with `GXNewText`, you must specify its contents and position on the screen.

In Listing 3-1, the value of `myPoint` specifies where QuickDraw GX should display the text shape. The code creates the text shape with the call to `GXNewText`. The text shape starts out with the default style, but the call to `GXSetShapeTextSize` changes the

Text Shapes

text size from the default to 120 points. This sample uses the `ff` macro, a shorthand notation for the `IntToFixed` macro. Both versions of the macro are described in the “Mathematics” chapter in *Inside Macintosh: QuickDraw GX Environment and Utilities*.

The `GXNewShape` function is described in the “Shape Objects” chapter in *Inside Macintosh: QuickDraw GX Objects*.

Listing 3-1 Creating a text shape with a nondefault text size

```
gxPoint myPoint = {ff(50), ff(150)};
gxShape myTextShape = GXNewText(4, (unsigned char*)"Wow!",
&myPoint);
GXSetShapeTextSize(myTextShape, ff(120));
```

The function `GXNewText` is described on page 3-8. The function `GXSetShapeTextSize` is described in the chapter “Typographic Styles” in this book.

To draw an existing text shape, use the `GXDrawShape` function.

You can also use the `GXDrawText` function to create, draw, and dispose of a text shape. The function uses the text shape’s default style object to determine what the font, text size, and other style attributes of the shape should be. This function is useful if you only need to draw the shape once and if you don’t need to set any characteristics of the style used.

```
GXDrawText(myTextLen, myText, myPosition);
```

The `GXDrawText` function is described on page 3-9. The `GXDrawShape` function is described in the “Shape Objects” chapter in *Inside Macintosh: QuickDraw GX Objects*.

Changing Text in a Text Shape

If you want to change all of the text of an existing shape, use the `GXSetText` function, described on page 3-12. This function takes an existing shape, a character count, a string of text, and a position. You can replace the text in the shape, change the position, or both. (If you want to replace only one of these elements, you pass `nil` for the other.)

If you want to change only some of the text of an existing text shape, use the `GXSetTextParts` function, described on page 3-14. The function takes an existing text shape, a glyph index corresponding to a glyph in that shape, a number of characters to be replaced, a number of characters to be added, and the new text to be inserted. You can insert new text while maintaining all the current text, or you can replace or delete existing text from the shape. Table 3-1 lists some of the uses of the `GXSetTextParts` function.

Table 3-1 Changing text in a text shape using the `GXSetTextParts` function

Action	Index	Old character count	New character count	Text
Inserting new text	Glyph index at which new text should start or <code>gxSelectToEnd</code> , if you want to insert at the end	0	Length of the new text	Pointer to the new text
Replacing and deleting some text	Glyph index at which the new text should start	Number of glyphs to delete from the original shape or <code>gxSelectToEnd</code>	Length of the new text	Pointer to the new text
Replacing all text in the shape	1	<code>gxSelectToEnd</code>	Length of the new text	Pointer to the new text
Deleting all text from the shape	1	<code>gxSelectToEnd</code>	0	<code>nil</code>

For example, suppose you want to change a text shape that reads “The dog” so that it reads “The beast”. The index value, which is the location of the first glyph to replace, is 5. The old character count is 3 (which corresponds to the number of glyphs in the word “dog”), and the new character count is 5 (the number of glyphs in “beast”). You could also set the old character count to the `gxSelectToEnd` constant, because you want to replace all of the text from position 5 to the end of the text in the shape. Listing 3-2 shows how to make this change.

Listing 3-2 Replacing text in a text shape

```
char      *myString = "The dog";
char      *newWord  = "beast";
short     textLength, insertLength;
gxPoint   myPoint = {ff(20), ff(20)};

textLength = strlen(myString);
insertLength = strlen(newWord);
gShape = GXNewText(textLength, (unsigned char *) myString,
                  &myPoint);
GXSetTextParts(gShape, 5, gxSelectToEnd, insertLength,
              (unsigned char *) newWord);
```

Text Shapes Reference

Functions

This section describes the functions that you use for creating, drawing, or changing QuickDraw GX text shapes.

Creating and Drawing Text Shapes

You can create a text shape using the `GXNewText` function. If you simply want to draw some text without creating a new shape, you can use the `GXDrawText` function instead.

GXNewText

You can use the `GXNewText` function to create a text shape.

```
gxShape GXNewText(long charCount, const unsigned char text[],
                  const gxPoint *position);
```

<code>charCount</code>	The number of character codes in the <code>text</code> parameter. For non-Roman scripts, the actual byte length may be up to twice the number of characters. If the value of <code>charCount</code> is 0, the text shape is equivalent to the empty shape.
<code>text</code>	An array of text data. The value of this parameter may be <code>nil</code> if the value of <code>charCount</code> is 0.
<code>position</code>	A pointer to the position, in geometry coordinates, of the starting point of the shape. This position is the intersection of the baseline with the left margin of the left-side bearing at the first glyph in the shape. If you pass <code>nil</code> , <code>GXNewText</code> sets the position to (0.0,0.0).

function result The new text shape.

DESCRIPTION

The `GXNewText` function creates a copy of the default text shape, sets the owner count of the copy to 1, initializes its geometry with the values in the function's parameters, and returns a reference to it as the function result.

The new text shape returned by this function contains references to the same style, ink, and transform objects as the default text shape.

Text Shapes

The parameter `charCount` indicates the number of characters present, which may not necessarily equal the number of bytes in the length of the `text` parameter. The interpretation of characters depends on the default text shape's style attribute.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>out_of_memory</code>	
<code>parameter_is_nil</code>	(debugging version)
<code>count_is_less_than_zero</code>	(debugging version)

SEE ALSO

To create a new glyph shape, use the `GXNewGlyphs` function, described in the chapter “Glyph Shapes” in this book.

To create a new layout shape, use the `GXNewLayout` function, described in the chapter “Layout Shapes” in this book.

For information about the default ink and transform objects, see *Inside Macintosh: QuickDraw GX Objects*.

The default values of the style object for any of the typographic shapes is discussed in the chapter “Typographic Styles” in this book.

You can determine the platform of the style object associated with the default shape using the `GXGetShapeEncoding` function, also described in the chapter “Typographic Styles.”

The default text shape is described on page 3-4.

GXDrawText

You can use the `GXDrawText` function to draw a string of text without first creating a text shape.

```
void GXDrawText(long charCount, const unsigned char text[],
                const gxPoint *position);
```

charCount The number of characters in the `text` parameter. For non-Roman scripts, the byte length may be up to twice the number of characters.

text An array of text data.

position A pointer to the position, in geometry coordinates, of the starting point of the shape. This is the intersection of the baseline with the left margin of the left-side bearing at the first glyph in the shape. If you pass `nil`, `GXDrawText` sets the position to (0.0,0.0).

Text Shapes

DESCRIPTION

The `GXDrawText` function draws the text string, starting at the point specified by `position`. The `charCount` parameter specifies the number of characters in the text drawn. The default text shape's style object specifies the font, text size, typestyle, baseline direction, and so on.

The parameter `charCount` indicates the number of characters present, which may not necessarily equal the number of bytes in the length of the `text` parameter. The interpretation of characters depends on the text shape's style attribute.

If the value of the `charCount` parameter is 0, the `GXDrawText` function does nothing.

You should use the `GXDrawShape` function if you want to draw the text contained in the `text` parameter several times (by creating a text shape) or if you want to draw the text using a style other than the default style.

ERRORS, WARNINGS, AND NOTICES

Errors

`parameter_is_nil`

(debugging version)

SEE ALSO

To draw an existing text shape, use the `GXDrawShape` function, described in the chapter “Shape Objects” in *Inside Macintosh: QuickDraw GX Objects*.

To draw a line of text that contains glyphs of different fonts, text sizes, or typestyles, use of the `GXDrawGlyphs` function, described in the chapter “Glyph Shapes” in this book.

To draw a line of text using the unique characteristics of the layout shape, see the description of the `GXDrawLayout` function in the chapter “Layout Shapes” in this book.

Manipulating Geometries of Text Shapes

You can obtain the values in a text shape's geometry—the number of character codes it contains, the character codes themselves, or the shape's position—using the `GXGetText` function. You can set any of these values, including replacing all of the text of the shape, by using the `GXSetText` function.

If you want to retrieve part of the text contained in the shape, use the `GXGetTextParts` function. If you want to change only some of the text in a text shape, use the `GXSetTextParts` function.

GXGetText

You can use the `GXGetText` function to return the information in a text shape's geometry, such as its text string or its position.

```
long GXGetText(gxShape source, long *charCount,
               unsigned char text[], gxPoint *position);
```

<code>source</code>	A reference to the text shape whose character code values you want to change.
<code>charCount</code>	A pointer to a long value. On return, the number of characters in the shape specified by <code>source</code> . If you pass <code>nil</code> in this parameter, <code>GXGetText</code> does not return a value.
<code>text</code>	A character array. On return, it contains the text string from the source shape. You must allocate the memory for this string. If you pass <code>nil</code> in this parameter, <code>GXGetText</code> does not return the text string.
<code>position</code>	A pointer to a point structure. On return, the point is the position, in geometry coordinates, of the text shape. If you pass <code>nil</code> , <code>GXGetText</code> does not return the position.

function result The number of bytes of text in the text shape.

DESCRIPTION

The `GXGetText` function returns the number of characters, the text string, the position of a text shape, and (in the function result) the byte length of the text shape specified by `source`. The `charCount` parameter may not be equal to the function result; `charCount` indicates the number of characters in the shape, which may not necessarily equal the number of bytes. Call this function twice, once to determine the size of the text array (pass `nil` for `text`), and a second time to fill out the array.

ERRORS, WARNINGS, AND NOTICES

Errors

`shape_is_nil`
`illegal_type_for_shape` (debugging version)

SEE ALSO

To get information from the geometry of a glyph shape, use the `GXGetGlyphs` function, described in the chapter “Glyph Shapes” in this book.

To get information from the geometry of a layout shape, use the `GXGetLayout` function, described in the chapter “Layout Shapes” in this book.

GXSetText

You can use the `GXSetText` function to insert a new text string into a text shape, or to change the text shape's position, or both.

```
void GXSetText(gxShape target, long charCount,
               const unsigned char text[],
               const gxPoint *position);
```

<code>target</code>	A reference to the text shape whose character code values you want to change.
<code>charCount</code>	The number of characters to be copied into the new text shape.
<code>text</code>	A character array containing the text to be copied into the text shape.
<code>position</code>	A pointer to a point structure specifying the location of the text shape, in geometry coordinates.

DESCRIPTION

The `GXSetText` function replaces the geometry of the specified shape with the text in the `text` parameter and the point in the `position` parameter. The shape type is set to `gxTextType`.

If the value of `charCount` is 0 and the value of `text` is not `nil`, the number of bytes in the existing text shape determines the length of the text copied.

The parameter `charCount` indicates the number of characters present, which may not necessarily equal the number of bytes in the length of the `text` parameter. The interpretation of characters depends on the text shape's style attribute.

You can shorten the shape by passing `nil` for the `text` parameter and a new character count for the text shape. QuickDraw GX changes the contents of the text shape to however many glyphs you specify. You cannot lengthen a text shape using this method, because QuickDraw GX returns the warning `new_shape_contains_invalid_data`.

If you don't want to change one of the values in the shape, such as the text or the position, set that parameter to `nil`. If you want to set the value of either the text or the position to `nil`, pass the value `gxSetToNil` in that parameter.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>shape_is_nil</code>	
<code>count_is_less_than_zero</code>	(debugging version)
<code>size_of_text_exceeds_implementation_limit</code>	(debugging version)
<code>out_of_memory</code>	(debugging version)

Warnings

<code>shape_access_not_allowed</code>	(debugging version)
<code>new_shape_contains_invalid_data</code>	(debugging version)

Notices (debugging version)`text_already_set`**SEE ALSO**

For more information on how to use the `GXSetText` function, see “Changing Text in a Text Shape” beginning on page 3-6.

You can use the `GXSetTextParts` function, described on page 3-14, to replace part of the text in a text shape or to insert new text into the shape.

You can also change the position of a text shape using the `GXMoveShape` or `GXMoveShapeTo` function, described in the “Transform Objects” chapter in *Inside Macintosh: QuickDraw GX Objects*.

To replace the geometry of a glyph shape with new text, use the `GXSetGlyphs` function, described in the chapter “Glyph Shapes” in this book.

To replace the geometry of a layout shape with new text, use the `GXSetLayout` function, described in the chapter “Layout Shapes” in this book.

To change a shape of another type into a text shape, use the `GXSetShapeType` function, described in the “Shape Objects” chapter of *Inside Macintosh: QuickDraw GX Objects*.

GXGetTextParts

You can use the `GXGetTextParts` function to retrieve part of the text of a text shape.

```
long GXGetTextParts(gxShape source, long index, long charCount,
                   unsigned char text[]);
```

<code>source</code>	A reference to the shape from which you retrieve data.
<code>index</code>	The index of the first glyph that you want to retrieve. This value must be greater than or equal to 1.
<code>charCount</code>	The number of character codes you want to retrieve. This value must be greater than or equal to 1, or it can be <code>gxSelectToEnd</code> , which selects all glyphs, beginning at the glyph index specified in <code>index</code> .
<code>text</code>	A pointer to a character array. On return, the array contains the text retrieved from the source shape.

function result The number of bytes of the retrieved text.

Text Shapes

DESCRIPTION

The `GXGetTextParts` function retrieves the specified number of character codes from the shape. The `text` parameter is a pointer to the retrieved text. You can retrieve some or all of the character codes from the shape using this function.

The shape specified in the `source` parameter must be of type `gxTextType`. If it isn't, `GXGetTextParts` posts the error `illegal_type_for_shape`.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>shape_is_nil</code>	
<code>index_is_less_than_one</code>	(debugging version)
<code>count_is_less_than_one</code>	(debugging version)
<code>illegal_type_for_shape</code>	(debugging version)

Warnings

<code>index_out_of_range</code>
<code>count_out_of_range</code>

SEE ALSO

You can use the `GXGetText` function (page 3-11) to retrieve all of the text of a text shape.

To retrieve part of the text from a glyph shape, use the `GXGetGlyphParts` function, described in the chapter “Glyph Shapes” in this book.

To retrieve part of the text from a layout shape, use the `GXGetLayoutParts` function described in the chapter “Layout Shapes” in this book.

GXSetTextParts

You can use the `GXSetTextParts` function to change part of the text of a text shape.

```
void GXSetTextParts(gxShape target, long index, long oldCharCount,
                   long newCharCount, const unsigned char text[]);
```

<code>target</code>	A reference to the text shape whose character code values you want to change.
<code>index</code>	The index of the first glyph where the editing operation will begin. If you are deleting text, this is the first glyph deleted; if you are inserting text, the new glyphs will appear before this one. This value must be greater than or equal to 0. A value of <code>gxSelectToEnd</code> in the <code>index</code> parameter indicates that <code>GXSetTextParts</code> should add the new glyphs after the last glyph in the text string. A value of 1 indicates the beginning of the text of the shape.

Text Shapes

`oldCharCount`

The number of glyphs you want to replace. This value can be greater than or equal to 0 (which indicates that you want to insert text), or it can be `gxSelectToEnd` (which is equal to -1 and indicates that you want to replace the text from the position specified by `index` to the end of the available text).

`newCharCount`

The number of character codes you want to add to the text shape.

`text`

A pointer to a character array containing the new text that you want to put into the text shape.

DESCRIPTION

The `GXSetTextParts` function replaces the text in the target shape with the text pointed to by the `text` parameter. The function inserts the new text at the glyph index specified by the `index` parameter. If the value of `oldCharCount` is greater than 0, the function replaces a corresponding number of glyphs in the original text string with a number of glyphs from the new text string. The number of new glyphs added is specified by the `newCharCount` parameter. If the value of the `oldCharCount` parameter is 0, the function inserts the new text but does not delete any of the original text.

The target shape must be of type `gxTextType`. If it isn't, `GXSetTextParts` posts the error `illegal_type_for_shape`.

ERRORS, WARNINGS, AND NOTICES

Errors

<code>shape_is_nil</code>	
<code>parameter_out_of_range</code>	(debugging version)
<code>inconsistent_parameters</code>	(debugging version)
<code>index_is_less_than_zero</code>	(debugging version)
<code>count_is_less_than_zero</code>	(debugging version)
<code>illegal_type_for_shape</code>	(debugging version)

Warnings

<code>index_out_of_range</code>	
<code>count_out_of_range</code>	
<code>shape_access_not_allowed</code>	(debugging version)

SEE ALSO

For more information on how to use the `GXSetTextParts` function, see “Changing Text in a Text Shape” beginning on page 3-6.

To replace all of the text in a text shape, you can use the `GXSetText` function, described on page 3-12.

To replace glyphs in a glyph shape, use the `GXSetGlyphParts` function, described in the chapter “Glyph Shapes” in this book.

To replace glyphs in a layout shape, use the `GXSetLayoutParts` function, described in the chapter “Layout Shapes” in this book.

Summary of Text Shapes

Functions

Creating and Drawing Text Shapes

```
gxShape GXNewText          (long charCount, const unsigned char text[],
                             const gxPoint *position);

void GXDrawText            (long charCount, const unsigned char text[],
                             const gxPoint *position);
```

Manipulating Geometries of Text Shapes

```
long GXGetText             (gxShape source, long *charCount, unsigned char
                             text[], gxPoint *position)

void GXSetText             (gxShape target, long charCount, const unsigned
                             char text[], const gxPoint *position);

long GXGetTextParts        (gxShape source, long index, long charCount,
                             unsigned char text[]);

void GXSetTextParts        (gxShape target, long index, long oldCharCount,
                             long newCharCount, const unsigned char text[]);
```