

SpeedText 2

Function Reference

Copyright © Christian Klaussner

1. Functions

Function **TextInitialize**(buffer)

Initializes the SpeedText library. This function needs to be called before any other function of this library. Otherwise all other functions will fail.

Parameter *buffer* indicates the default rendering buffer. Usually **BackBuffer()** is used for the default buffer. You can change it after initialization by calling **TextSetBuffer**.

Returns *True* if initialization was successful, otherwise *False*.

Function **TextDeinitialize**()

Deinitializes the SpeedText library and deletes all of its resources (Except fonts that have been created using **TextLoadFont**. You should delete them using **TextFreeFont** before deinitialization).

Returns *True* if deinitialization was successful, otherwise *False*.

Function **TextSetBuffer**(buffer)

Sets the current rendering buffer. This buffer will be the target for **TextDraw** and **TextDrawRect**.

Function **TextGetBuffer**()

Returns the current rendering buffer.

Function **TextFreeFont**(font)

Deletes the specified font. After calling this function you can no longer use the font handle.

Function **TextSetFont**(font)

Sets the current font for drawing and metrics computations such as **TextStringWidth** and **TextFontWidth**. Specify *Zero* for *font* to use the default font (Courier).

Function **TextGetFont()**

Returns the currently selected font. If no font was selected using **TextSetFont**, the return value is *Zero*.

Function **TextLoadFont**(fontname\$, height, bold, italic, underline, quality, filename\$)

Loads a font with the specified characteristics. Parameter *fontname\$* indicates the name of the font, for example „Times New Roman“. If the font is not installed on the system, but available as a TTF or OTF file, you can specify the filename for parameter *filename\$*. Note that the true name of the font still needs to be specified.

Parameter *height* determines the font height in pixels. Parameters *bold*, *italic* and *underline* can be set to *True* or *False*.

Parameter *quality* can be one of the following values:

TEXT_DEFAULT	Use the quality settings used by the operating system.
TEXT_NONANTIALIASED	Don't use anti-aliasing.
TEXT_ANTIALIASED	Use anti-aliasing.
TEXT_CLEARTYPE	Use ClearType anti-aliasing.

Returns the new font handle if loading was successful, otherwise *Zero*.

Function **TextDraw**(x, y, text\$, ax, ay, encoding)

Draws text into the currently selected buffer at position (*x*, *y*). Parameter *ax* and *ay* indicate the alignment for the respective axes.

Values for horizontal alignment:

TEXT_LEFT	The reference point is on the left edge of the text.
TEXT_CENTER	Text is horizontally centered at the reference point.
TEXT_RIGHT	The reference point is on the right edge of the text.

Values for vertical alignment:

TEXT_TOP	The reference point is on the top edge of the text.
TEXT_MIDDLE	Text is vertically centered at the reference point.
TEXT_BOTTOM	The reference point is on the bottom edge of the text.

For valid encoding types, see chapter **Text Encoding**.

Function **TextDrawRect**(x, y, width, height, text\$, ax, format, encoding)

Draws text into the currently selected buffer using a rectangle and formatting options. The formatting is based on the rectangle specified by *width* and *height*. If you don't know the size of the rectangle specify -1 for both *width* and *height*.

Parameter *ax* can be either TEXT_LEFT, TEXT_CENTER or TEXT_RIGHT. See **TextDraw** for details.

Parameter *format* can be one of the following values:

TEXT_WORDWRAP	Lines are broken if a word does not fit into the rectangle.
TEXT_DONTCLIP	Text that does not fit into the rectangle won't be clipped.

For valid encoding types, see chapter **Text Encoding**.

Function **TextLockBuffer**()

Locks the currently selected buffer for faster text rendering. A buffer can only be used for text rendering using **TextDraw** and **TextDrawRect** while it is locked.

Function **TextUnlockBuffer**()

Unlocks the currently selected buffer.

Function **TextFontWidth**()

Returns the width of the broadest character in the currently selected font.

Function **TextFontHeight**()

Returns the height of the highest character in the currently selected font.

Function **TextFontAscent**()

Returns the ascent of the currently selected font. The sum of ascent and descent of a font is equal to its height.

Function **TextFontDescent()**

Returns the descent of the currently selected font. The sum of ascent and descent of a font is equal to its height.

Function **TextStringWidth**(text\$, encoding)

Returns the width of the string *text*. The currently selected font is used for measurement.

For valid encoding types, see chapter **Text Encoding**.

Function **TextStringHeight**(text\$, encoding)

Returns the height of the string *text*. The currently selected font is used for measurement.

For valid encoding types, see chapter **Text Encoding**.

Function **TextSetColor**(red, green, blue)

Sets the color for text rendering in RGB format.

Function **TextSetBackground**(red, green, blue)

Sets the background color for text rendering in RGB format. Specify -1 for all colors to make the background transparent.

Function **TextColorRed()**

Returns the red component of the currently selected color.

Function **TextColorGreen()**

Returns the green component of the currently selected color.

Function **TextColorBlue()**

Returns the blue component of the currently selected color.

Function **TextBackgroundRed()**

Returns the red component of the currently selected background color.

Function **TextBackgroundGreen()**

Returns the green component of the currently selected background color.

Function **TextBackgroundBlue()**

Returns the blue component of the currently selected background color.

2. Text Encoding

Text can be encoded using ANSI or Unicode (UTF-8). The standard Blitz3D IDE uses ANSI strings that are adequate for most of the western languages like English or German. If you need Asian or Arabic text, you have to save it in UTF-8 format using a suitable editor like Notepad. All SpeedText functions that accept text strings as a parameter have an additional parameter *encoding*. You can specify either TEXT_ANSI or TEXT_UTF8, depending on how your text is encoded.

3. Performance Issues

A performance problem exists when using SpeedText with Windows Vista. Drawing performance of standard Blitz3D text is faster than SpeedText in most cases due to Windows Vista's graphics architecture. This problem does not occur on Windows XP and Windows 7 RC.