FTGL

2.1.3~rc5

Generated by Doxygen 1.5.6

Thu Jun 12 14:45:00 2008

Contents

Chapter 1

FTGL User Guide

1.1 Introduction

OpenGL doesn't provide direct font support, so the application must use any of OpenGL's other features for font rendering, such as drawing bitmaps or pixmaps, creating texture maps containing an entire character set, drawing character outlines, or creating a 3D geometry for each character.

More information can be found on the OpenGL website:

- `http://www.opengl.org/resources/faq/technical/fonts.htm`
- `http://www.opengl.org/resources/features/fontsurvey/`

Most of these systems require a pre-processing stage to take the native fonts and convert them into a proprietary format.

FTGL was born out of the need to treat fonts in OpenGL applications just like any other application. For example when using Adobe Photoshop or Microsoft Word you don't need an intermediate pre-processing step to use high quality scalable fonts.

1.2 Documentation

- **FTGL tutorial** (p. **??**)
- C API reference:
  - **FTGlyph.h** (p. 92)

  - **FTFont.h** (p. 76)
  - **FTLayout.h** (p. 95)

- C++ API reference:
  - class **FTGlyph** (p. 35)
  - class **FTFont** (p. 24)
  - class **FTLayout** (p. 38)

1.3 Additional information

- **Frequently Asked Questions** (p. **??**)
- **Projects using FTGL** (p. **??**)

Chapter 2

Namespace Documentation

2.1 FTGL Namespace Reference

Enumerations

- enum **RenderMode** { **RENDER_FRONT** = 0x0001, **RENDER_BACK** = 0x0002, **RENDER_-SIDE** = 0x0004, **RENDER_ALL** = 0xffff }
- enum **TextAlignment** { **ALIGN_LEFT** = 0, **ALIGN_CENTER** = 1, **ALIGN_RIGHT** = 2, **ALIGN_JUSTIFY** = 3 }

2.1.1 Enumeration Type Documentation

2.1.1.1 enum FTGL::RenderMode

Enumerator:

RENDER_FRONT

RENDER_BACK

RENDER_SIDE

RENDER_ALL

Definition at line 53 of file ftgl.h.

2.1.1.2 enum FTGL::TextAlignment

Enumerator:

ALIGN_LEFT

ALIGN_CENTER

ALIGN_RIGHT

ALIGN_JUSTIFY

Definition at line 61 of file ftgl.h.

Chapter 3

Data Structure Documentation

3.1 FTBBox Class Reference

```
#include <FTBBox.h>
```

3.1.1 Detailed Description

FTBBox (p. 5) is a convenience class for handling bounding boxes.

Definition at line 42 of file FTBBox.h.

Public Member Functions

- **FTBBox** ()

  Default constructor.

- **FTBBox** (float lx, float ly, float lz, float ux, float uy, float uz)

  Constructor.

- **FTBBox** (**FTPoint** l, **FTPoint** u)

  Constructor.

- **FTBBox** (FT_GlyphSlot glyph)

  Constructor.

- ∼**FTBBox** ()

  Destructor.

- void **Invalidate** ()

  Mark the bounds invalid by setting all lower dimensions greater than the upper dimensions.

- bool **IsValid** ()

  Determines if this bounding box is valid.

- **FTBBox** & **operator+=** (const **FTPoint** vector)

Move the Bounding Box by a vector.

- **FTBBox** & **operator|=** (const **FTBBox** &bbox)

  Combine two bounding boxes.

- void **SetDepth** (float depth)
- **FTPoint** const **Upper** () const
- **FTPoint** const **Lower** () const

3.1.2 Constructor & Destructor Documentation

3.1.2.1 FTBBox::FTBBox () `[inline]`

Default constructor.

Bounding box is set to zero.

Definition at line 48 of file FTBBox.h.

3.1.2.2 FTBBox::FTBBox (float *lx*, float *ly*, float *lz*, float *ux*, float *uy*, float *uz*) `[inline]`

Constructor.

Definition at line 56 of file FTBBox.h.

3.1.2.3 FTBBox::FTBBox (FTPoint *l*, FTPoint *u*) `[inline]`

Constructor.

Definition at line 64 of file FTBBox.h.

3.1.2.4 FTBBox::FTBBox (FT_GlyphSlot *glyph*) `[inline]`

Constructor.

Extracts a bounding box from a freetype glyph. Uses the control box for the glyph. `FT_Glyph_Get_CBox()`

Parameters:

glyph A freetype glyph

Definition at line 75 of file FTBBox.h.

3.1.2.5 FTBBox::∼FTBBox () `[inline]`

Destructor.

Definition at line 93 of file FTBBox.h.

3.1.3 Member Function Documentation

3.1.3.1 void FTBBox::Invalidate () `[inline]`

Mark the bounds invalid by setting all lower dimensions greater than the upper dimensions.

Definition at line 100 of file FTBBox.h.

3.1.3.2 bool FTBBox::IsValid () `[inline]`

Determines if this bounding box is valid.

Returns:

True if all lower values are <= the corresponding upper values.

Definition at line 112 of file FTBBox.h.

3.1.3.3 FTBBox& FTBBox::operator+= (const FTPoint *vector*) `[inline]`

Move the Bounding Box by a vector.

Parameters:

vector The vector to move the bbox in 3D space.

Definition at line 124 of file FTBBox.h.

3.1.3.4 FTBBox& FTBBox::operator|= (const FTBBox & *bbox*) `[inline]`

Combine two bounding boxes.

The result is the smallest bounding box containing the two original boxes.

Parameters:

bbox The bounding box to merge with the second one.

Definition at line 138 of file FTBBox.h.

References lower, upper, FTPoint::X(), FTPoint::Y(), and FTPoint::Z().

3.1.3.5 void FTBBox::SetDepth (float *depth*) `[inline]`

Definition at line 150 of file FTBBox.h.

3.1.3.6 FTPoint const FTBBox::Upper () const `[inline]`

Definition at line 159 of file FTBBox.h.

Referenced by FTFont::BBox().

3.1.3.7 FTPoint const FTBBox::Lower () const `[inline]`

Definition at line 165 of file FTBBox.h.

Referenced by FTFont::BBox().

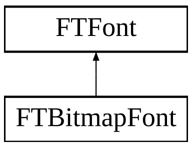
The documentation for this class was generated from the following file:

- **FTBBox.h**

3.2 FTBitmapFont Class Reference

```
#include <FTGLBitmapFont.h>
```

Inheritance diagram for FTBitmapFont::



3.2.1 Detailed Description

FTBitmapFont (p. 9) is a specialisation of the **FTFont** (p. 24) class for handling Bitmap fonts.

See also:

FTFont (p. 24)

Definition at line 45 of file FTGLBitmapFont.h.

Public Member Functions

- **FTBitmapFont** (const char ∗fontFilePath)
  Open and read a font file.
- **FTBitmapFont** (const unsigned char ∗pBufferBytes, size_t bufferSizeInBytes)
  Open and read a font from a buffer in memory.
- ∼**FTBitmapFont** ()
  Destructor.

Protected Member Functions

- virtual **FTGlyph** ∗ **MakeGlyph** (FT_GlyphSlot slot)
  Construct a glyph of the correct type.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 FTBitmapFont::FTBitmapFont (const char ∗ *fontFilePath*)

Open and read a font file.

Sets Error flag.

Parameters:

fontFilePath font file path.

3.2.2.2 FTBitmapFont::FTBitmapFont (const unsigned char ∗ *pBufferBytes*, size_t *bufferSizeInBytes*)

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by **FTGL** (p. 3). The pointer must be valid while using **FTGL** (p. 3).

Parameters:

pBufferBytes the in-memory buffer

bufferSizeInBytes the length of the buffer in bytes

3.2.2.3 FTBitmapFont::∼FTBitmapFont ()

Destructor.

3.2.3 Member Function Documentation

3.2.3.1 virtual FTGlyph∗ FTBitmapFont::MakeGlyph (FT_GlyphSlot *slot*) `[protected, virtual]`

Construct a glyph of the correct type.

Clients must override the function and return their specialised **FTGlyph** (p. 35).

Parameters:

slot A FreeType glyph slot.

Returns:

An FT∗∗∗∗Glyph or `null` on failure.

Implements **FTFont** (p. 33).

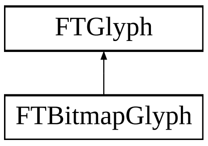
The documentation for this class was generated from the following file:

- **FTGLBitmapFont.h**

3.3 FTBitmapGlyph Class Reference

`#include <FTBitmapGlyph.h>`

Inheritance diagram for FTBitmapGlyph::



3.3.1 Detailed Description

FTBitmapGlyph (p. 11) is a specialisation of **FTGlyph** (p. 35) for creating bitmaps.

Definition at line 42 of file FTBitmapGlyph.h.

Public Member Functions

- **FTBitmapGlyph** (FT_GlyphSlot glyph)
  Constructor.
- virtual ∼**FTBitmapGlyph** ()
  Destructor.
- virtual const **FTPoint** & **Render** (const **FTPoint** &pen, int renderMode)
  Render this glyph at the current pen position.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 FTBitmapGlyph::FTBitmapGlyph (FT_GlyphSlot *glyph*)

Constructor.

Parameters:

glyph The Freetype glyph to be processed

3.3.2.2 virtual FTBitmapGlyph::∼FTBitmapGlyph () `[virtual]`

Destructor.

3.3.3 Member Function Documentation

3.3.3.1 virtual const FTPoint& FTBitmapGlyph::Render (const FTPoint & *pen*, int *renderMode*) `[virtual]`

Render this glyph at the current pen position.

Parameters:

pen The current pen position.

renderMode Render mode to display

Returns:

The advance distance for this glyph.

Implements **FTGlyph** (p. 36).

The documentation for this class was generated from the following file:

- **FTBitmapGlyph.h**

3.4 FTBuffer Class Reference

```
#include <FTBuffer.h>
```

3.4.1 Detailed Description

FTBuffer (p. 13) is a helper class for pixel buffers.

It provides the interface between **FTBufferFont** (p. 16) and **FTBufferGlyph** (p. 18) to optimise rendering operations.

See also:

FTBufferGlyph (p. 18)
FTBufferFont (p. 16)

Definition at line 45 of file FTBuffer.h.

Public Member Functions

- **FTBuffer** ()
  Default constructor.
- ~**FTBuffer** ()
  Destructor.
- **FTPoint Pos** () const
  Get the pen's position in the buffer.
- void **Pos** (**FTPoint** arg)
  Set the pen's position in the buffer.
- void **Size** (int w, int h)
  Set the buffer's size.
- int **Width** () const
  Get the buffer's width.
- int **Height** () const
  Get the buffer's height.
- unsigned char ∗ **Pixels** () const
  Get the buffer's direct pixel buffer.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 FTBuffer::FTBuffer ()

Default constructor.

3.4.2.2 FTBuffer::∼FTBuffer ()

Destructor.

3.4.3 Member Function Documentation

3.4.3.1 FTPoint FTBuffer::Pos () const `[inline]`

Get the pen's position in the buffer.

Returns:

The pen's position as an **FTPoint** (p. 49) object.

Definition at line 63 of file FTBuffer.h.

3.4.3.2 void FTBuffer::Pos (FTPoint *arg*) `[inline]`

Set the pen's position in the buffer.

Parameters:

arg An **FTPoint** (p. 49) object with the desired pen's position.

Definition at line 73 of file FTBuffer.h.

3.4.3.3 void FTBuffer::Size (int *w*, int *h*)

Set the buffer's size.

Parameters:

w The buffer's desired width, in pixels.
h The buffer's desired height, in pixels.

3.4.3.4 int FTBuffer::Width () const `[inline]`

Get the buffer's width.

Returns:

The buffer's width, in pixels.

Definition at line 91 of file FTBuffer.h.

3.4.3.5 int FTBuffer::Height () const `[inline]`

Get the buffer's height.

Returns:

The buffer's height, in pixels.

Definition at line 98 of file FTBuffer.h.

3.4.3.6 unsigned char∗ FTBuffer::Pixels () const `[inline]`

Get the buffer's direct pixel buffer.

Returns:

A read-write pointer to the buffer's pixels.

Definition at line 105 of file FTBuffer.h.

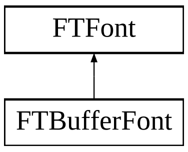
The documentation for this class was generated from the following file:

- **FTBuffer.h**

3.5 FTBufferFont Class Reference

`#include <FTBufferFont.h>`

Inheritance diagram for FTBufferFont::



3.5.1 Detailed Description

FTBufferFont (p. 16) is a specialisation of the **FTFont** (p. 24) class for handling memory buffer fonts.

See also:

FTFont (p. 24)

Definition at line 43 of file FTBufferFont.h.

Public Member Functions

- **FTBufferFont** (const char ∗fontFilePath)
  Open and read a font file.
- **FTBufferFont** (const unsigned char ∗pBufferBytes, size_t bufferSizeInBytes)
  Open and read a font from a buffer in memory.
- ∼**FTBufferFont** ()
  Destructor.

Protected Member Functions

- virtual **FTGlyph** ∗ **MakeGlyph** (FT_GlyphSlot slot)
  Construct a glyph of the correct type.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 FTBufferFont::FTBufferFont (const char ∗ *fontFilePath*)

Open and read a font file.

Sets Error flag.

Parameters:

fontFilePath font file path.

3.5.2.2 FTBufferFont::FTBufferFont (const unsigned char * *pBufferBytes*, size_t *bufferSizeInBytes*)

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by **FTGL** (p. 3). The pointer must be valid while using **FTGL** (p. 3).

Parameters:

pBufferBytes the in-memory buffer

bufferSizeInBytes the length of the buffer in bytes

3.5.2.3 FTBufferFont::∼FTBufferFont ()

Destructor.

3.5.3 Member Function Documentation

3.5.3.1 virtual FTGlyph* FTBufferFont::MakeGlyph (FT_GlyphSlot *slot*) `[protected, virtual]`

Construct a glyph of the correct type.

Clients must override the function and return their specialised **FTGlyph** (p. 35).

Parameters:

slot A FreeType glyph slot.

Returns:

An FT****Glyph or `null` on failure.

Implements **FTFont** (p. 33).

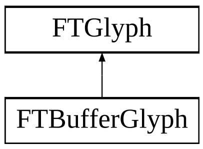
The documentation for this class was generated from the following file:

- **FTBufferFont.h**

3.6 FTBufferGlyph Class Reference

`#include <FTBufferGlyph.h>`

Inheritance diagram for FTBufferGlyph::



3.6.1 Detailed Description

FTBufferGlyph (p. 18) is a specialisation of **FTGlyph** (p. 35) for memory buffer rendering.

Definition at line 40 of file FTBufferGlyph.h.

Public Member Functions

- **FTBufferGlyph** (FT_GlyphSlot glyph, **FTBuffer** ∗buffer)

  Constructor.

- virtual ∼**FTBufferGlyph** ()

  Destructor.

- virtual const **FTPoint** & **Render** (const **FTPoint** &pen, int renderMode)

  Render this glyph at the current pen position.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 FTBufferGlyph::FTBufferGlyph (FT_GlyphSlot *glyph*, FTBuffer ∗ *buffer*)

Constructor.

Parameters:

glyph The Freetype glyph to be processed

buffer An **FTBuffer** (p. 13) object in which to render the glyph.

3.6.2.2 virtual FTBufferGlyph::∼FTBufferGlyph () `[virtual]`

Destructor.

3.6.3 Member Function Documentation

3.6.3.1 virtual const FTPoint& FTBufferGlyph::Render (const FTPoint & *pen*, int *renderMode*) `[virtual]`

Render this glyph at the current pen position.

Parameters:

pen The current pen position.

renderMode Render mode to display

Returns:

The advance distance for this glyph.

Implements **FTGlyph** (p. 36).

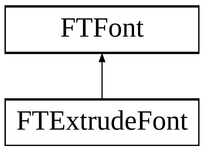
The documentation for this class was generated from the following file:

- **FTBufferGlyph.h**

3.7 FTExtrudeFont Class Reference

```
#include <FTGLExtrdFont.h>
```

Inheritance diagram for FTExtrudeFont::



3.7.1 Detailed Description

FTExtrudeFont (p. 20) is a specialisation of the **FTFont** (p. 24) class for handling extruded Polygon fonts.

See also:

FTFont (p. 24)
FTPolygonFont (p. 56)

Definition at line 46 of file FTGLExtrdFont.h.

Public Member Functions

- **FTExtrudeFont** (const char ∗fontFilePath)

  Open and read a font file.

- **FTExtrudeFont** (const unsigned char ∗pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.

- ∼**FTExtrudeFont** ()

  Destructor.

Protected Member Functions

- virtual **FTGlyph** ∗ **MakeGlyph** (FT_GlyphSlot slot)

  Construct a glyph of the correct type.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 FTExtrudeFont::FTExtrudeFont (const char ∗ *fontFilePath*)

Open and read a font file.

Sets Error flag.

Parameters:

fontFilePath font file path.

3.7.2.2 FTExtrudeFont::FTExtrudeFont (const unsigned char ∗ *pBufferBytes*, size_t *bufferSizeInBytes*)

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by **FTGL** (p. 3). The pointer must be valid while using **FTGL** (p. 3).

Parameters:

pBufferBytes the in-memory buffer

bufferSizeInBytes the length of the buffer in bytes

3.7.2.3 FTExtrudeFont::∼FTExtrudeFont ()

Destructor.

3.7.3 Member Function Documentation

3.7.3.1 virtual FTGlyph∗ FTExtrudeFont::MakeGlyph (FT_GlyphSlot *slot*) `[protected, virtual]`

Construct a glyph of the correct type.

Clients must override the function and return their specialised **FTGlyph** (p. 35).

Parameters:

slot A FreeType glyph slot.

Returns:

An FT∗∗∗∗Glyph or `null` on failure.

Implements **FTFont** (p. 33).

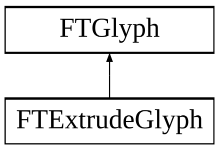
The documentation for this class was generated from the following file:

- **FTGLExtrdFont.h**

3.8 FTExtrudeGlyph Class Reference

`#include <FTExtrdGlyph.h>`

Inheritance diagram for FTExtrudeGlyph::



3.8.1 Detailed Description

FTExtrudeGlyph (p. 22) is a specialisation of **FTGlyph** (p. 35) for creating tessellated extruded polygon glyphs.

Definition at line 43 of file FTExtrdGlyph.h.

Public Member Functions

- **FTExtrudeGlyph** (FT_GlyphSlot glyph, float depth, float frontOutset, float backOutset, bool useDisplayList)

  Constructor.

- virtual ~**FTExtrudeGlyph** ()

  Destructor.

- virtual const **FTPoint** & **Render** (const **FTPoint** &pen, int renderMode)

  Render this glyph at the current pen position.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 FTExtrudeGlyph::FTExtrudeGlyph (FT_GlyphSlot *glyph*, float *depth*, float *frontOutset*, float *backOutset*, bool *useDisplayList*)

Constructor.

Sets the Error to Invalid_Outline if the glyph isn't an outline.

Parameters:

glyph The Freetype glyph to be processed

depth The distance along the z axis to extrude the glyph

frontOutset outset contour size

backOutset outset contour size

useDisplayList Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

3.8.2.2 virtual FTExtrudeGlyph::∼FTExtrudeGlyph () `[virtual]`

Destructor.

3.8.3 Member Function Documentation

3.8.3.1 virtual const FTPoint& FTExtrudeGlyph::Render (const FTPoint & *pen*, int *renderMode*) `[virtual]`

Render this glyph at the current pen position.

Parameters:

pen The current pen position.

renderMode Render mode to display

Returns:

The advance distance for this glyph.

Implements **FTGlyph** (p. 36).

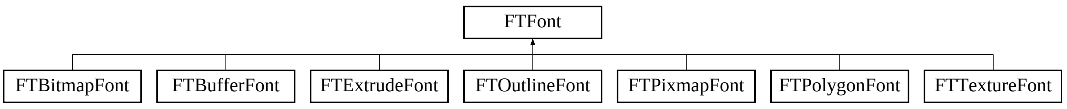
The documentation for this class was generated from the following file:

- **FTExtrdGlyph.h**

3.9 FTFont Class Reference

`#include <FTFont.h>`

Inheritance diagram for FTFont::



3.9.1 Detailed Description

FTFont (p. 24) is the public interface for the **FTGL** (p. 3) library.

Specific font classes are derived from this class. It uses the helper classes FTFace and FTSize to access the Freetype library. This class is abstract and deriving classes must implement the protected `MakeGlyph` function to create glyphs of the appropriate type.

It is good practice after using these functions to test the error code returned. `FT_Error` **`Error()`** `(p. 33)`. Check the freetype file fterrdef.h for error definitions.

See also:

FTFace
FTSize

Definition at line 56 of file FTFont.h.

Public Member Functions

- virtual ∼**FTFont** ()
- virtual bool **Attach** (const char ∗fontFilePath)

  Attach auxilliary file to font e.g font metrics.

- virtual bool **Attach** (const unsigned char ∗pBufferBytes, size_t bufferSizeInBytes)

  Attach auxilliary data to font e.g font metrics, from memory.

- virtual void **GlyphLoadFlags** (FT_Int flags)

  Set the glyph loading flags.

- virtual bool **CharMap** (FT_Encoding encoding)

  Set the character map for the face.

- virtual unsigned int **CharMapCount** () const

  Get the number of character maps in this face.

- virtual FT_Encoding ∗ **CharMapList** ()

  Get a list of character maps in this face.

- virtual bool **FaceSize** (const unsigned int size, const unsigned int res=72)

Set the char size for the current face.

- virtual unsigned int **FaceSize** () const

  Get the current face size in points (1/72 inch).

- virtual void **Depth** (float depth)

  Set the extrusion distance for the font.

- virtual void **Outset** (float outset)

  Set the outset distance for the font.

- virtual void **Outset** (float front, float back)

  Set the front and back outset distances for the font.

- virtual void **UseDisplayList** (bool useList)

  *Enable or disable the use of Display Lists inside **FTGL** (*p. *3).*

- virtual float **Ascender** () const

  Get the global ascender height for the face.

- virtual float **Descender** () const

  Gets the global descender height for the face.

- virtual float **LineHeight** () const

  Gets the line spacing for the font.

- virtual **FTBBox BBox** (const char ∗string, const int len=-1, **FTPoint** position=**FTPoint**(), **FTPoint** spacing=**FTPoint**())

  Get the bounding box for a string.

- void **BBox** (const char ∗string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).

- virtual **FTBBox BBox** (const wchar_t ∗string, const int len=-1, **FTPoint** position=**FTPoint**(), **FT-Point** spacing=**FTPoint**())

  Get the bounding box for a string.

- void **BBox** (const wchar_t ∗string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).

- virtual float **Advance** (const char ∗string, const int len=-1, **FTPoint** spacing=**FTPoint**())

  Get the advance for a string.

- virtual float **Advance** (const wchar_t ∗string, const int len=-1, **FTPoint** spacing=**FTPoint**())

  Get the advance for a string.

- virtual **FTPoint Render** (const char ∗string, const int len=-1, **FTPoint** position=**FTPoint**(), **FT-Point** spacing=**FTPoint**(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.

- virtual **FTPoint Render** (const wchar_t ∗string, const int len=-1, **FTPoint** position=**FTPoint**(), **FT-Point** spacing=**FTPoint**(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.

- virtual FT_Error **Error** () const

  Queries the Font for errors.

Protected Member Functions

- **FTFont** (char const ∗fontFilePath)

  Open and read a font file.

- **FTFont** (const unsigned char ∗pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.

- virtual **FTGlyph** ∗ **MakeGlyph** (FT_GlyphSlot slot)=0

  Construct a glyph of the correct type.

Friends

- class **FTBitmapFont**
- class **FTBufferFont**
- class **FTExtrudeFont**
- class **FTOutlineFont**
- class **FTPixmapFont**
- class **FTPolygonFont**
- class **FTTextureFont**
- class **FTFontImpl**

3.9.2 Constructor & Destructor Documentation

3.9.2.1 FTFont::FTFont (char const ∗ *fontFilePath*) `[protected]`

Open and read a font file.

Sets Error flag.

Parameters:

fontFilePath	font file path.

3.9.2.2 FTFont::FTFont (const unsigned char ∗ *pBufferBytes*, size_t *bufferSizeInBytes*) `[protected]`

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by **FTGL** (p. 3). The pointer must be valid while using **FTGL** (p. 3).

Parameters:

pBufferBytes the in-memory buffer

bufferSizeInBytes the length of the buffer in bytes

3.9.2.3 virtual FTFont::∼FTFont () `[virtual]`

3.9.3 Member Function Documentation

3.9.3.1 virtual bool FTFont::Attach (const char ∗ *fontFilePath*) `[virtual]`

Attach auxilliary file to font e.g font metrics.

Note: not all font formats implement this function.

Parameters:

fontFilePath auxilliary font file path.

Returns:

`true` if file has been attached successfully.

3.9.3.2 virtual bool FTFont::Attach (const unsigned char ∗ *pBufferBytes*, size_t *bufferSizeInBytes*) `[virtual]`

Attach auxilliary data to font e.g font metrics, from memory.

Note: not all font formats implement this function.

Parameters:

pBufferBytes the in-memory buffer.

bufferSizeInBytes the length of the buffer in bytes.

Returns:

`true` if file has been attached successfully.

3.9.3.3 virtual void FTFont::GlyphLoadFlags (FT_Int *flags*) `[virtual]`

Set the glyph loading flags.

By default, fonts use the most sensible flags when loading a font's glyph using FT_Load_Glyph(). This function allows to override the default flags.

Parameters:

flags The glyph loading flags.

3.9.3.4 virtual bool FTFont::CharMap (FT_Encoding *encoding*) `[virtual]`

Set the character map for the face.

Parameters:

encoding Freetype enumerate for char map code.

Returns:

`true` if charmap was valid and set correctly.

3.9.3.5 virtual unsigned int FTFont::CharMapCount () const `[virtual]`

Get the number of character maps in this face.

Returns:

character map count.

3.9.3.6 virtual FT_Encoding∗ FTFont::CharMapList () `[virtual]`

Get a list of character maps in this face.

Returns:

pointer to the first encoding.

3.9.3.7 virtual bool FTFont::FaceSize (const unsigned int *size*, const unsigned int *res* = `72`) `[virtual]`

Set the char size for the current face.

Parameters:

size the face size in points (1/72 inch)
res the resolution of the target device.

Returns:

`true` if size was set correctly

3.9.3.8 virtual unsigned int FTFont::FaceSize () const `[virtual]`

Get the current face size in points (1/72 inch).

Returns:

face size

3.9.3.9 virtual void FTFont::Depth (float *depth*) `[virtual]`

Set the extrusion distance for the font.

Only implemented by **FTExtrudeFont** (p. 20)

Parameters:

depth The extrusion distance.

3.9.3.10 virtual void FTFont::Outset (float *outset*) `[virtual]`

Set the outset distance for the font.

Only implemented by **FTOutlineFont** (p. 41), **FTPolygonFont** (p. 56) and **FTExtrudeFont** (p. 20)

Parameters:

outset The outset distance.

3.9.3.11 virtual void FTFont::Outset (float *front*, float *back*) `[virtual]`

Set the front and back outset distances for the font.

Only implemented by **FTExtrudeFont** (p. 20)

Parameters:

front The front outset distance.

back The back outset distance.

3.9.3.12 virtual void FTFont::UseDisplayList (bool *useList*) `[virtual]`

Enable or disable the use of Display Lists inside **FTGL** (p. 3).

Parameters:

useList `true` turns ON display lists. `false` turns OFF display lists.

3.9.3.13 virtual float FTFont::Ascender () const `[virtual]`

Get the global ascender height for the face.

Returns:

Ascender height

3.9.3.14 virtual float FTFont::Descender () const `[virtual]`

Gets the global descender height for the face.

Returns:

Descender height

3.9.3.15 virtual float FTFont::LineHeight () const `[virtual]`

Gets the line spacing for the font.

Returns:

Line height

3.9.3.16 virtual FTBBox FTFont::BBox (const char ∗ *string*, const int *len* = `-1`, FTPoint *position* = FTPoint `()`, FTPoint *spacing* = FTPoint `()`) `[virtual]`

Get the bounding box for a string.

Parameters:

string A char buffer.

len The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).

position The pen position of the first character (optional).

spacing A displacement vector to add after each character has been checked (optional).

Returns:

The corresponding bounding box.

Referenced by BBox().

3.9.3.17 void FTFont::BBox (const char ∗ *string*, float & *llx*, float & *lly*, float & *llz*, float & *urx*, float & *ury*, float & *urz*) `[inline]`

Get the bounding box for a string (deprecated).

Parameters:

string A char buffer.

llx Lower left near x coordinate.

lly Lower left near y coordinate.

llz Lower left near z coordinate.

urx Upper right far x coordinate.

ury Upper right far y coordinate.

urz Upper right far z coordinate.

Definition at line 251 of file FTFont.h.

References BBox(), FTBBox::Lower(), FTBBox::Upper(), FTPoint::Xf(), FTPoint::Yf(), and FT-Point::Zf().

3.9.3.18 virtual FTBBox FTFont::BBox (const wchar_t ∗ *string*, const int *len* = `-1`, FTPoint *position* = FTPoint `()`, FTPoint *spacing* = FTPoint `()`) `[virtual]`

Get the bounding box for a string.

Parameters:

string A wchar_t buffer.

len The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).

position The pen position of the first character (optional).

spacing A displacement vector to add after each character has been checked (optional).

Returns:

The corresponding bounding box.

3.9.3.19 void FTFont::BBox (const wchar_t ∗ *string*, float & *llx*, float & *lly*, float & *llz*, float & *urx*, float & *ury*, float & *urz*) `[inline]`

Get the bounding box for a string (deprecated).

Parameters:

string A wchar_t buffer.

llx Lower left near x coordinate.

lly Lower left near y coordinate.

llz Lower left near z coordinate.

urx Upper right far x coordinate.

ury Upper right far y coordinate.

urz Upper right far z coordinate.

Definition at line 286 of file FTFont.h.

References BBox(), FTBBox::Lower(), FTBBox::Upper(), FTPoint::Xf(), FTPoint::Yf(), and FTPoint::Zf().

3.9.3.20 virtual float FTFont::Advance (const char ∗ *string*, const int *len* = `-1`, FTPoint *spacing* = FTPoint `()`) `[virtual]`

Get the advance for a string.

Parameters:

string 'C' style string to be checked.

len The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).

spacing A displacement vector to add after each character has been checked (optional).

Returns:

The string's advance width.

3.9.3.21 virtual float FTFont::Advance (const wchar_t ∗ *string*, const int *len* = `-1`, FTPoint *spacing* = FTPoint `()`) `[virtual]`

Get the advance for a string.

Parameters:

string A wchar_t string

len The length of the string. If <0 then all characters will be checked until a null character is encountered (optional).

spacing A displacement vector to add after each character has been checked (optional).

Returns:

The string's advance width.

3.9.3.22 virtual FTPoint FTFont::Render (const char ∗ *string*, const int *len* = `-1`, FTPoint *position* = FTPoint `()`, FTPoint *spacing* = FTPoint `()`, int *renderMode* = `FTGL::RENDER_ALL`) `[virtual]`

Render a string of characters.

Parameters:

string 'C' style string to be output.

len The length of the string. If <0 then all characters will be displayed until a null character is encountered (optional).

position The pen position of the first character (optional).

spacing A displacement vector to add after each character has been displayed (optional).

renderMode Render mode to use for display (optional).

Returns:

The new pen position after the last character was output.

3.9.3.23 virtual FTPoint FTFont::Render (const wchar_t ∗ *string*, const int *len* = `-1`, FTPoint *position* = FTPoint `()`, FTPoint *spacing* = FTPoint `()`, int *renderMode* = `FTGL::RENDER_ALL`) `[virtual]`

Render a string of characters.

Parameters:

string wchar_t string to be output.

len The length of the string. If <0 then all characters will be displayed until a null character is encountered (optional).

position The pen position of the first character (optional).

spacing A displacement vector to add after each character has been displayed (optional).

renderMode Render mode to use for display (optional).

Returns:

The new pen position after the last character was output.

3.9.3.24 virtual FT_Error FTFont::Error () const `[virtual]`

Queries the Font for errors.

Returns:

The current error code.

3.9.3.25 virtual FTGlyph* FTFont::MakeGlyph (FT_GlyphSlot *slot*) `[protected, pure virtual]`

Construct a glyph of the correct type.

Clients must override the function and return their specialised **FTGlyph** (p. 35).

Parameters:

slot A FreeType glyph slot.

Returns:

An FT∗∗∗∗Glyph or `null` on failure.

Implemented in **FTBufferFont** (p. 17), **FTBitmapFont** (p. 10), **FTExtrudeFont** (p. 21), **FTOutlineFont** (p. 42), **FTPixmapFont** (p. 46), **FTPolygonFont** (p. 57), and **FTTextureFont** (p. 65).

3.9.4 Friends And Related Function Documentation

3.9.4.1 friend class FTBitmapFont `[friend]`

Definition at line 78 of file FTFont.h.

3.9.4.2 friend class FTBufferFont `[friend]`

Definition at line 79 of file FTFont.h.

3.9.4.3 friend class FTExtrudeFont `[friend]`

Definition at line 80 of file FTFont.h.

3.9.4.4 friend class FTOutlineFont `[friend]`

Definition at line 81 of file FTFont.h.

3.9.4.5 friend class FTPixmapFont `[friend]`

Definition at line 82 of file FTFont.h.

3.9.4.6 friend class FTPolygonFont `[friend]`

Definition at line 83 of file FTFont.h.

3.9.4.7 friend class FTTextureFont `[friend]`

Definition at line 84 of file FTFont.h.

3.9.4.8 friend class FTFontImpl `[friend]`

Definition at line 367 of file FTFont.h.

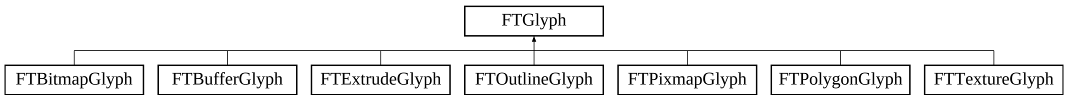
The documentation for this class was generated from the following file:

- **FTFont.h**

3.10 FTGlyph Class Reference

```
#include <FTGlyph.h>
```

Inheritance diagram for FTGlyph::



3.10.1 Detailed Description

FTGlyph (p. 35) is the base class for **FTGL** (p. 3) glyphs.

It provides the interface between Freetype glyphs and their openGL renderable counterparts. This is an abstract class and derived classes must implement the `Render` function.

See also:

FTBBox (p. 5)
FTPoint (p. 49)

Definition at line 50 of file FTGlyph.h.

Public Member Functions

- virtual ∼**FTGlyph** ()

  Destructor.

- virtual const **FTPoint** & **Render** (const **FTPoint** &pen, int renderMode)=0

  Renders this glyph at the current pen position.

- virtual float **Advance** () const

  Return the advance width for this glyph.

- virtual const **FTBBox** & **BBox** () const

  Return the bounding box for this glyph.

- virtual FT_Error **Error** () const

  Queries for errors.

Protected Member Functions

- **FTGlyph** (FT_GlyphSlot glyph)

  Create a glyph.

Friends

- class **FTBitmapGlyph**
- class **FTBufferGlyph**
- class **FTExtrudeGlyph**
- class **FTOutlineGlyph**
- class **FTPixmapGlyph**
- class **FTPolygonGlyph**
- class **FTTextureGlyph**

3.10.2 Constructor & Destructor Documentation

3.10.2.1 FTGlyph::FTGlyph (FT_GlyphSlot *glyph*) `[protected]`

Create a glyph.

Parameters:

glyph The Freetype glyph to be processed

3.10.2.2 virtual FTGlyph::∼FTGlyph () `[virtual]`

Destructor.

3.10.3 Member Function Documentation

3.10.3.1 virtual const FTPoint& FTGlyph::Render (const FTPoint & *pen*, int *renderMode*) `[pure virtual]`

Renders this glyph at the current pen position.

Parameters:

pen The current pen position.

renderMode Render mode to display

Returns:

The advance distance for this glyph.

Implemented in **FTBitmapGlyph** (p. 11), **FTBufferGlyph** (p. 19), **FTExtrudeGlyph** (p. 23), **FTOutlineGlyph** (p. 44), **FTPixmapGlyph** (p. 47), **FTPolygonGlyph** (p. 59), and **FTTextureGlyph** (p. 67).

3.10.3.2 virtual float FTGlyph::Advance () const `[virtual]`

Return the advance width for this glyph.

Returns:

advance width.

3.10.3.3 virtual const FTBBox& FTGlyph::BBox () const `[virtual]`

Return the bounding box for this glyph.

Returns:

bounding box.

3.10.3.4 virtual FT_Error FTGlyph::Error () const `[virtual]`

Queries for errors.

Returns:

The current error code.

3.10.4 Friends And Related Function Documentation

3.10.4.1 friend class FTBitmapGlyph `[friend]`

Definition at line 70 of file FTGlyph.h.

3.10.4.2 friend class FTBufferGlyph `[friend]`

Definition at line 71 of file FTGlyph.h.

3.10.4.3 friend class FTExtrudeGlyph `[friend]`

Definition at line 72 of file FTGlyph.h.

3.10.4.4 friend class FTOutlineGlyph `[friend]`

Definition at line 73 of file FTGlyph.h.

3.10.4.5 friend class FTPixmapGlyph `[friend]`

Definition at line 74 of file FTGlyph.h.

3.10.4.6 friend class FTPolygonGlyph `[friend]`

Definition at line 75 of file FTGlyph.h.

3.10.4.7 friend class FTTextureGlyph `[friend]`

Definition at line 76 of file FTGlyph.h.

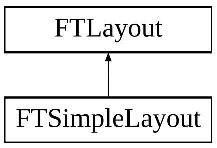
The documentation for this class was generated from the following file:

- **FTGlyph.h**

3.11 FTLayout Class Reference

`#include <FTLayout.h>`

Inheritance diagram for FTLayout::



3.11.1 Detailed Description

FTLayout (p. 38) is the interface for layout managers that render text.

Specific layout manager classes are derived from this class. This class is abstract and deriving classes must implement the protected `Render` methods to render formatted text and `BBox` methods to determine the bounding box of output text.

See also:

FTFont (p. 24)
FTBBox (p. 5)

Definition at line 52 of file FTLayout.h.

Public Member Functions

- virtual ∼**FTLayout** ()

  Destructor.

- virtual **FTBBox BBox** (const char ∗string, const int len=-1, **FTPoint** position=**FTPoint**())=0

  Get the bounding box for a formatted string.

- virtual **FTBBox BBox** (const wchar_t ∗string, const int len=-1, **FTPoint** position=**FTPoint**())=0

  Get the bounding box for a formatted string.

- virtual void **Render** (const char ∗string, const int len=-1, **FTPoint** position=**FTPoint**(), int renderMode=FTGL::RENDER_ALL)=0

  Render a string of characters.

- virtual void **Render** (const wchar_t ∗string, const int len=-1, **FTPoint** position=**FTPoint**(), int renderMode=FTGL::RENDER_ALL)=0

  Render a string of characters.

- virtual FT_Error **Error** () const

  Queries the Layout for errors.

Protected Member Functions

- **FTLayout** ()

Friends

- class **FTSimpleLayout**

3.11.2 Constructor & Destructor Documentation

3.11.2.1 FTLayout::FTLayout () `[protected]`

3.11.2.2 virtual FTLayout::∼FTLayout () `[virtual]`

Destructor.

3.11.3 Member Function Documentation

3.11.3.1 virtual FTBBox FTLayout::BBox (const char ∗ *string*, const int *len* = `-1`, FTPoint *position* = FTPoint`()`) `[pure virtual]`

Get the bounding box for a formatted string.

Parameters:

string A char string.

len The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).

position The pen position of the first character (optional).

Returns:

The corresponding bounding box.

Implemented in **FTSimpleLayout** (p. 61).

3.11.3.2 virtual FTBBox FTLayout::BBox (const wchar_t ∗ *string*, const int *len* = `-1`, FTPoint *position* = FTPoint`()`) `[pure virtual]`

Get the bounding box for a formatted string.

Parameters:

string A wchar_t string.

len The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).

position The pen position of the first character (optional).

Returns:

The corresponding bounding box.

Implemented in **FTSimpleLayout** (p. 61).

3.11.3.3 virtual void FTLayout::Render (const char * *string*, const int *len* = `-1`, FTPoint *position* = FTPoint `()`, int *renderMode* = `FTGL::RENDER_ALL`) `[pure virtual]`

Render a string of characters.

Parameters:

string 'C' style string to be output.

len The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).

position The pen position of the first character (optional).

renderMode Render mode to display (optional)

Implemented in **FTSimpleLayout** (p. 62).

3.11.3.4 virtual void FTLayout::Render (const wchar_t * *string*, const int *len* = `-1`, FTPoint *position* = FTPoint `()`, int *renderMode* = `FTGL::RENDER_ALL`) `[pure virtual]`

Render a string of characters.

Parameters:

string wchar_t string to be output.

len The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).

position The pen position of the first character (optional).

renderMode Render mode to display (optional)

Implemented in **FTSimpleLayout** (p. 62).

3.11.3.5 virtual FT_Error FTLayout::Error () const `[virtual]`

Queries the Layout for errors.

Returns:

The current error code.

3.11.4 Friends And Related Function Documentation

3.11.4.1 friend class FTSimpleLayout `[friend]`

Definition at line 67 of file FTLayout.h.

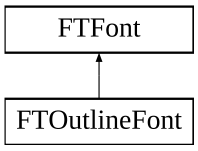
The documentation for this class was generated from the following file:

- **FTLayout.h**

3.12 FTOutlineFont Class Reference

`#include <FTGLOutlineFont.h>`

Inheritance diagram for FTOutlineFont::



3.12.1 Detailed Description

FTOutlineFont (p. 41) is a specialisation of the **FTFont** (p. 24) class for handling Vector Outline fonts.

See also:

FTFont (p. 24)

Definition at line 45 of file FTGLOutlineFont.h.

Public Member Functions

- **FTOutlineFont** (const char ∗fontFilePath)
  Open and read a font file.
- **FTOutlineFont** (const unsigned char ∗pBufferBytes, size_t bufferSizeInBytes)
  Open and read a font from a buffer in memory.
- ∼**FTOutlineFont** ()
  Destructor.

Protected Member Functions

- virtual **FTGlyph** ∗ **MakeGlyph** (FT_GlyphSlot slot)
  Construct a glyph of the correct type.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 FTOutlineFont::FTOutlineFont (const char ∗ *fontFilePath*)

Open and read a font file.

Sets Error flag.

Parameters:

fontFilePath font file path.

3.12.2.2 FTOutlineFont::FTOutlineFont (const unsigned char ∗ *pBufferBytes*, size_t *bufferSizeInBytes*)

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by **FTGL** (p. 3). The pointer must be valid while using **FTGL** (p. 3).

Parameters:

pBufferBytes the in-memory buffer

bufferSizeInBytes the length of the buffer in bytes

3.12.2.3 FTOutlineFont::∼FTOutlineFont ()

Destructor.

3.12.3 Member Function Documentation

3.12.3.1 virtual FTGlyph∗ FTOutlineFont::MakeGlyph (FT_GlyphSlot *slot*) `[protected, virtual]`

Construct a glyph of the correct type.

Clients must override the function and return their specialised **FTGlyph** (p. 35).

Parameters:

slot A FreeType glyph slot.

Returns:

An FT∗∗∗∗Glyph or `null` on failure.

Implements **FTFont** (p. 33).

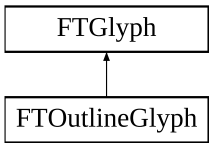
The documentation for this class was generated from the following file:

- **FTGLOutlineFont.h**

3.13 FTOutlineGlyph Class Reference

`#include <FTOutlineGlyph.h>`

Inheritance diagram for FTOutlineGlyph::



3.13.1 Detailed Description

FTOutlineGlyph (p. 43) is a specialisation of **FTGlyph** (p. 35) for creating outlines.

Definition at line 42 of file FTOutlineGlyph.h.

Public Member Functions

- **FTOutlineGlyph** (FT_GlyphSlot glyph, float outset, bool useDisplayList)

  Constructor.

- virtual ∼**FTOutlineGlyph** ()

  Destructor.

- virtual const **FTPoint** & **Render** (const **FTPoint** &pen, int renderMode)

  Render this glyph at the current pen position.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 FTOutlineGlyph::FTOutlineGlyph (FT_GlyphSlot *glyph*, float *outset*, bool *useDisplayList*)

Constructor.

Sets the Error to Invalid_Outline if the glyphs isn't an outline.

Parameters:

glyph The Freetype glyph to be processed

outset outset distance

useDisplayList Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

3.13.2.2 virtual FTOutlineGlyph::∼FTOutlineGlyph () `[virtual]`

Destructor.

3.13.3 Member Function Documentation

3.13.3.1 virtual const FTPoint& FTOutlineGlyph::Render (const FTPoint & *pen*, int *renderMode*) `[virtual]`

Render this glyph at the current pen position.

Parameters:

pen The current pen position.

renderMode Render mode to display

Returns:

The advance distance for this glyph.

Implements **FTGlyph** (p. 36).

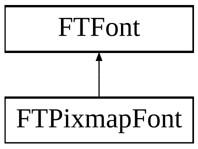
The documentation for this class was generated from the following file:

- **FTOutlineGlyph.h**

3.14 FTPixmapFont Class Reference

`#include <FTGLPixmapFont.h>`

Inheritance diagram for FTPixmapFont::



3.14.1 Detailed Description

FTPixmapFont (p. 45) is a specialisation of the **FTFont** (p. 24) class for handling Pixmap (Grey Scale) fonts.

See also:

FTFont (p. 24)

Definition at line 45 of file FTGLPixmapFont.h.

Public Member Functions

- **FTPixmapFont** (const char ∗fontFilePath)
  Open and read a font file.
- **FTPixmapFont** (const unsigned char ∗pBufferBytes, size_t bufferSizeInBytes)
  Open and read a font from a buffer in memory.
- ∼**FTPixmapFont** ()
  Destructor.

Protected Member Functions

- virtual **FTGlyph** ∗ **MakeGlyph** (FT_GlyphSlot slot)
  Construct a glyph of the correct type.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 FTPixmapFont::FTPixmapFont (const char ∗ *fontFilePath*)

Open and read a font file.

Sets Error flag.

Parameters:

fontFilePath font file path.

3.14.2.2 FTPixmapFont::FTPixmapFont (const unsigned char ∗ *pBufferBytes*, size_t *bufferSizeInBytes*)

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by **FTGL** (p. 3). The pointer must be valid while using **FTGL** (p. 3).

Parameters:

pBufferBytes the in-memory buffer

bufferSizeInBytes the length of the buffer in bytes

3.14.2.3 FTPixmapFont::∼FTPixmapFont ()

Destructor.

3.14.3 Member Function Documentation

3.14.3.1 virtual FTGlyph∗ FTPixmapFont::MakeGlyph (FT_GlyphSlot *slot*) `[protected, virtual]`

Construct a glyph of the correct type.

Clients must override the function and return their specialised **FTGlyph** (p. 35).

Parameters:

slot A FreeType glyph slot.

Returns:

An FT∗∗∗∗Glyph or `null` on failure.

Implements **FTFont** (p. 33).

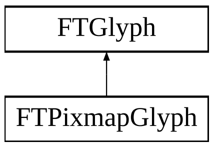
The documentation for this class was generated from the following file:

- **FTGLPixmapFont.h**

3.15 FTPixmapGlyph Class Reference

`#include <FTPixmapGlyph.h>`

Inheritance diagram for FTPixmapGlyph::



3.15.1 Detailed Description

FTPixmapGlyph (p. 47) is a specialisation of **FTGlyph** (p. 35) for creating pixmaps.

Definition at line 42 of file FTPixmapGlyph.h.

Public Member Functions

- **FTPixmapGlyph** (FT_GlyphSlot glyph)
  Constructor.
- virtual ~**FTPixmapGlyph** ()
  Destructor.
- virtual const **FTPoint** & **Render** (const **FTPoint** &pen, int renderMode)
  Render this glyph at the current pen position.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 FTPixmapGlyph::FTPixmapGlyph (FT_GlyphSlot *glyph*)

Constructor.

Parameters:

glyph The Freetype glyph to be processed

3.15.2.2 virtual FTPixmapGlyph::~FTPixmapGlyph () `[virtual]`

Destructor.

3.15.3 Member Function Documentation

3.15.3.1 virtual const FTPoint& FTPixmapGlyph::Render (const FTPoint & *pen*, int *renderMode*) `[virtual]`

Render this glyph at the current pen position.

Parameters:

pen The current pen position.

renderMode Render mode to display

Returns:

The advance distance for this glyph.

Implements **FTGlyph** (p. 36).

The documentation for this class was generated from the following file:

- **FTPixmapGlyph.h**

3.16 FTPoint Class Reference

```
#include <FTPoint.h>
```

3.16.1 Detailed Description

FTPoint (p. 49) class is a basic 3-dimensional point or vector.

Definition at line 42 of file FTPoint.h.

Public Member Functions

- **FTPoint** ()

  Default constructor.

- **FTPoint** (const **FTGL_DOUBLE** x, const **FTGL_DOUBLE** y, const **FTGL_DOUBLE** z=0)

  Constructor.

- **FTPoint** (const FT_Vector &ft_vector)

  Constructor.

- **FTPoint Normalise** ()

  Normalise a point's coordinates.

- **FTPoint** & **operator+=** (const **FTPoint** &point)

  Operator += In Place Addition.

- **FTPoint operator+** (const **FTPoint** &point) const

  Operator +.

- **FTPoint** & **operator-=** (const **FTPoint** &point)

  Operator -= In Place Substraction.

- **FTPoint operator-** (const **FTPoint** &point) const

  Operator -.

- **FTPoint operator∗** (double multiplier) const

  Operator ∗ Scalar multiplication.

- **FTPoint operator^** (const **FTPoint** &point)

  Operator ^ Vector product.

- **operator const FTGL_DOUBLE** ∗ () const

  Cast to FTGL_DOUBLE∗.

- void **X** (**FTGL_DOUBLE** x)

  Setters.

- void **Y** (**FTGL_DOUBLE** y)

- void **Z** (**FTGL_DOUBLE** z)
- **FTGL_DOUBLE X** () const

  Getters.

- **FTGL_DOUBLE Y** () const
- **FTGL_DOUBLE Z** () const
- **FTGL_FLOAT Xf** () const
- **FTGL_FLOAT Yf** () const
- **FTGL_FLOAT Zf** () const

Friends

- **FTPoint operator∗** (double multiplier, **FTPoint** &point)

  Operator ∗ Scalar multiplication.

- double **operator∗** (**FTPoint** &a, **FTPoint** &b)

  Operator ∗ Scalar product.

- bool **operator==** (const **FTPoint** &a, const **FTPoint** &b)

  Operator == Tests for equality.

- bool **operator!=** (const **FTPoint** &a, const **FTPoint** &b)

  Operator != Tests for non equality.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 FTPoint::FTPoint () `[inline]`

Default constructor.

Point is set to zero.

Definition at line 48 of file FTPoint.h.

3.16.2.2 FTPoint::FTPoint (const FTGL_DOUBLE *x*, const FTGL_DOUBLE *y*, const FTGL_DOUBLE *z* = `0`) `[inline]`

Constructor.

Z coordinate is set to zero if unspecified.

Parameters:

x First component

y Second component

z Third component

Definition at line 62 of file FTPoint.h.

3.16.2.3 FTPoint::FTPoint (const FT_Vector & *ft_vector*) `[inline]`

Constructor.

This converts an FT_Vector to an **FTPoint** (p. 49)

Parameters:

ft_vector A freetype vector

Definition at line 75 of file FTPoint.h.

3.16.3 Member Function Documentation

3.16.3.1 FTPoint FTPoint::Normalise ()

Normalise a point's coordinates.

If the coordinates are zero, the point is left untouched.

Returns:

A vector of norm one.

3.16.3.2 FTPoint& FTPoint::operator+= (const FTPoint & *point*) `[inline]`

Operator += In Place Addition.

Parameters:

point

Returns:

this plus point.

Definition at line 97 of file FTPoint.h.

References values.

3.16.3.3 FTPoint FTPoint::operator+ (const FTPoint & *point*) const `[inline]`

Operator +.

Parameters:

point

Returns:

this plus point.

Definition at line 112 of file FTPoint.h.

References values.

3.16.3.4 FTPoint& FTPoint::operator-= (const FTPoint & *point*) `[inline]`

Operator -= In Place Substraction.

Parameters:

point

Returns:

this minus point.

Definition at line 128 of file FTPoint.h.

References values.

3.16.3.5 FTPoint FTPoint::operator- (const FTPoint & *point*) const `[inline]`

Operator -.

Parameters:

point

Returns:

this minus point.

Definition at line 143 of file FTPoint.h.

References values.

3.16.3.6 FTPoint FTPoint::operator$*$ (double *multiplier*) const `[inline]`

Operator $*$ Scalar multiplication.

Parameters:

multiplier

Returns:

`this` multiplied by `multiplier`.

Definition at line 159 of file FTPoint.h.

References values.

3.16.3.7 FTPoint FTPoint::operator^ (const FTPoint & *point*) `[inline]`

Operator ^ Vector product.

Parameters:

point Second point

Returns:

this vector point.

Definition at line 204 of file FTPoint.h.

References values.

3.16.3.8 FTPoint::operator const FTGL_DOUBLE ∗ () const `[inline]`

Cast to FTGL_DOUBLE∗.

Definition at line 240 of file FTPoint.h.

3.16.3.9 void FTPoint::X (FTGL_DOUBLE *x*) `[inline]`

Setters.

Definition at line 249 of file FTPoint.h.

Referenced by FTBBox::operator|=().

3.16.3.10 void FTPoint::Y (FTGL_DOUBLE *y*) `[inline]`

Definition at line 250 of file FTPoint.h.

Referenced by FTBBox::operator|=().

3.16.3.11 void FTPoint::Z (FTGL_DOUBLE *z*) `[inline]`

Definition at line 251 of file FTPoint.h.

Referenced by FTBBox::operator|=().

3.16.3.12 FTGL_DOUBLE FTPoint::X () const `[inline]`

Getters.

Definition at line 257 of file FTPoint.h.

3.16.3.13 FTGL_DOUBLE FTPoint::Y () const `[inline]`

Definition at line 258 of file FTPoint.h.

3.16.3.14 FTGL_DOUBLE FTPoint::Z () const `[inline]`

Definition at line 259 of file FTPoint.h.

3.16.3.15 FTGL_FLOAT FTPoint::Xf () const `[inline]`

Definition at line 260 of file FTPoint.h.

Referenced by FTFont::BBox().

3.16.3.16 FTGL_FLOAT FTPoint::Yf () const `[inline]`

Definition at line 261 of file FTPoint.h.

Referenced by FTFont::BBox().

3.16.3.17 FTGL_FLOAT FTPoint::Zf () const `[inline]`

Definition at line 262 of file FTPoint.h.

Referenced by FTFont::BBox().

3.16.4 Friends And Related Function Documentation

3.16.4.1 FTPoint operator∗ (double *multiplier*, FTPoint & *point*) `[friend]`

Operator ∗ Scalar multiplication.

Parameters:

point
multiplier

Returns:

`multiplier` multiplied by `point`.

Definition at line 177 of file FTPoint.h.

3.16.4.2 double operator∗ (FTPoint & *a*, FTPoint & *b*) `[friend]`

Operator ∗ Scalar product.

Parameters:

a First vector.
b Second vector.

Returns:

`a.b` scalar product.

Definition at line 190 of file FTPoint.h.

3.16.4.3 bool operator== (const FTPoint & *a*, const FTPoint & *b*) `[friend]`

Operator == Tests for equality.

Parameters:

a
b

Returns:

true if a & b are equal

3.16.4.4 bool operator!= (const FTPoint & *a*, const FTPoint & *b*) `[friend]`

Operator != Tests for non equality.

Parameters:

> ***a***
>
> ***b***

Returns:

> true if a & b are not equal

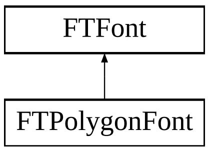
The documentation for this class was generated from the following file:

- **FTPoint.h**

3.17 FTPolygonFont Class Reference

```
#include <FTGLPolygonFont.h>
```

Inheritance diagram for FTPolygonFont::



3.17.1 Detailed Description

FTPolygonFont (p. 56) is a specialisation of the **FTFont** (p. 24) class for handling tesselated Polygon Mesh fonts.

See also:

FTFont (p. 24)

Definition at line 45 of file FTGLPolygonFont.h.

Public Member Functions

- **FTPolygonFont** (const char ∗fontFilePath)
  Open and read a font file.
- **FTPolygonFont** (const unsigned char ∗pBufferBytes, size_t bufferSizeInBytes)
  Open and read a font from a buffer in memory.
- ∼**FTPolygonFont** ()
  Destructor.

Protected Member Functions

- virtual **FTGlyph** ∗ **MakeGlyph** (FT_GlyphSlot slot)
  Construct a glyph of the correct type.

3.17.2 Constructor & Destructor Documentation

3.17.2.1 FTPolygonFont::FTPolygonFont (const char ∗ *fontFilePath*)

Open and read a font file.

Sets Error flag.

Parameters:

fontFilePath font file path.

3.17.2.2 FTPolygonFont::FTPolygonFont (const unsigned char ∗ *pBufferBytes*, size_t *bufferSizeInBytes*)

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by **FTGL** (p. 3). The pointer must be valid while using **FTGL** (p. 3).

Parameters:

pBufferBytes the in-memory buffer

bufferSizeInBytes the length of the buffer in bytes

3.17.2.3 FTPolygonFont::∼FTPolygonFont ()

Destructor.

3.17.3 Member Function Documentation

3.17.3.1 virtual FTGlyph∗ FTPolygonFont::MakeGlyph (FT_GlyphSlot *slot*) `[protected, virtual]`

Construct a glyph of the correct type.

Clients must override the function and return their specialised **FTGlyph** (p. 35).

Parameters:

slot A FreeType glyph slot.

Returns:

An FT∗∗∗∗Glyph or `null` on failure.

Implements **FTFont** (p. 33).

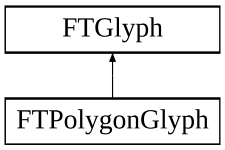
The documentation for this class was generated from the following file:

- **FTGLPolygonFont.h**

3.18 FTPolygonGlyph Class Reference

```
#include <FTPolyGlyph.h>
```

Inheritance diagram for FTPolygonGlyph::



3.18.1 Detailed Description

FTPolygonGlyph (p. 58) is a specialisation of **FTGlyph** (p. 35) for creating tessellated polygon glyphs.

Definition at line 43 of file FTPolyGlyph.h.

Public Member Functions

- **FTPolygonGlyph** (FT_GlyphSlot glyph, float outset, bool useDisplayList)

  Constructor.

- virtual ∼**FTPolygonGlyph** ()

  Destructor.

- virtual const **FTPoint** & **Render** (const **FTPoint** &pen, int renderMode)

  Render this glyph at the current pen position.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 FTPolygonGlyph::FTPolygonGlyph (FT_GlyphSlot *glyph*, float *outset*, bool *useDisplayList*)

Constructor.

Sets the Error to Invalid_Outline if the glyphs isn't an outline.

Parameters:

glyph The Freetype glyph to be processed

outset The outset distance

useDisplayList Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

3.18.2.2 virtual FTPolygonGlyph::∼FTPolygonGlyph () `[virtual]`

Destructor.

3.18.3 Member Function Documentation

3.18.3.1 virtual const FTPoint& FTPolygonGlyph::Render (const FTPoint & *pen*, int *renderMode*) `[virtual]`

Render this glyph at the current pen position.

Parameters:

pen The current pen position.

renderMode Render mode to display

Returns:

The advance distance for this glyph.

Implements **FTGlyph** (p. 36).

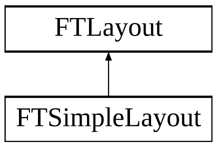
The documentation for this class was generated from the following file:

- **FTPolyGlyph.h**

3.19 FTSimpleLayout Class Reference

```
#include <FTSimpleLayout.h>
```

Inheritance diagram for FTSimpleLayout::



3.19.1 Detailed Description

FTSimpleLayout (p. 60) is a specialisation of **FTLayout** (p. 38) for simple text boxes.

This class has basic support for text wrapping, left, right and centered alignment, and text justification.

See also:

FTLayout (p. 38)

Definition at line 49 of file FTSimpleLayout.h.

Public Member Functions

- **FTSimpleLayout** ()

  Initializes line spacing to 1.0, alignment to ALIGN_LEFT and wrap to 100.0.

- ∼**FTSimpleLayout** ()

  Destructor.

- virtual **FTBBox BBox** (const char ∗string, const int len=-1, **FTPoint** position=**FTPoint**())

  Get the bounding box for a formatted string.

- virtual **FTBBox BBox** (const wchar_t ∗string, const int len=-1, **FTPoint** position=**FTPoint**())

  Get the bounding box for a formatted string.

- virtual void **Render** (const char ∗string, const int len=-1, **FTPoint** position=**FTPoint**(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.

- virtual void **Render** (const wchar_t ∗string, const int len=-1, **FTPoint** position=**FTPoint**(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.

- void **SetFont** (**FTFont** ∗fontInit)

  Set the font to use for rendering the text.

- **FTFont** ∗ **GetFont** ()

- void **SetLineLength** (const float LineLength)

  The maximum line length for formatting text.

- float **GetLineLength** () const
- void **SetAlignment** (const **FTGL::TextAlignment** Alignment)

  The text alignment mode used to distribute space within a line or rendered text.

- **FTGL::TextAlignment GetAlignment** () const
- void **SetLineSpacing** (const float LineSpacing)

  Sets the line height.

- float **GetLineSpacing** () const

3.19.2 Constructor & Destructor Documentation

3.19.2.1 FTSimpleLayout::FTSimpleLayout ()

Initializes line spacing to 1.0, alignment to ALIGN_LEFT and wrap to 100.0.

3.19.2.2 FTSimpleLayout::∼FTSimpleLayout ()

Destructor.

3.19.3 Member Function Documentation

3.19.3.1 virtual FTBBox FTSimpleLayout::BBox (const char ∗ *string*, const int *len* = `-1`, FTPoint *position* = FTPoint `()`) `[virtual]`

Get the bounding box for a formatted string.

Parameters:

string A char string.

len The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).

position The pen position of the first character (optional).

Returns:

The corresponding bounding box.

Implements **FTLayout** (p. 39).

3.19.3.2 virtual FTBBox FTSimpleLayout::BBox (const wchar_t ∗ *string*, const int *len* = `-1`, FTPoint *position* = FTPoint `()`) `[virtual]`

Get the bounding box for a formatted string.

Parameters:

string A wchar_t string.

len The length of the string. If <0 then all characters will be checked until a null character is encountered (optional).

position The pen position of the first character (optional).

Returns:

The corresponding bounding box.

Implements **FTLayout** (p. 39).

3.19.3.3 virtual void FTSimpleLayout::Render (const char ∗ *string*, const int *len* = `-1`, FTPoint *position* = FTPoint `()`, int *renderMode* = `FTGL::RENDER_ALL`) `[virtual]`

Render a string of characters.

Parameters:

string 'C' style string to be output.

len The length of the string. If <0 then all characters will be displayed until a null character is encountered (optional).

position The pen position of the first character (optional).

renderMode Render mode to display (optional)

Implements **FTLayout** (p. 40).

3.19.3.4 virtual void FTSimpleLayout::Render (const wchar_t ∗ *string*, const int *len* = `-1`, FTPoint *position* = FTPoint `()`, int *renderMode* = `FTGL::RENDER_ALL`) `[virtual]`

Render a string of characters.

Parameters:

string wchar_t string to be output.

len The length of the string. If <0 then all characters will be displayed until a null character is encountered (optional).

position The pen position of the first character (optional).

renderMode Render mode to display (optional)

Implements **FTLayout** (p. 40).

3.19.3.5 void FTSimpleLayout::SetFont (FTFont ∗ *fontInit*)

Set the font to use for rendering the text.

Parameters:

fontInit A pointer to the new font. The font is referenced by this but will not be disposed of when this is deleted.

3.19.3.6 FTFont∗ FTSimpleLayout::GetFont ()

Returns:

The current font.

3.19.3.7 void FTSimpleLayout::SetLineLength (const float *LineLength*)

The maximum line length for formatting text.

Parameters:

LineLength The new line length.

3.19.3.8 float FTSimpleLayout::GetLineLength () const

Returns:

The current line length.

3.19.3.9 void FTSimpleLayout::SetAlignment (const FTGL::TextAlignment *Alignment*)

The text alignment mode used to distribute space within a line or rendered text.

Parameters:

Alignment The new alignment mode.

3.19.3.10 FTGL::TextAlignment FTSimpleLayout::GetAlignment () const

Returns:

The text alignment mode.

3.19.3.11 void FTSimpleLayout::SetLineSpacing (const float *LineSpacing*)

Sets the line height.

Parameters:

LineSpacing The height of each line of text expressed as a percentage of the current fonts line height.

3.19.3.12 float FTSimpleLayout::GetLineSpacing () const

Returns:

The line spacing.

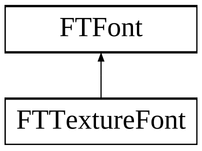
The documentation for this class was generated from the following file:

- **FTSimpleLayout.h**

3.20 FTTextureFont Class Reference

```
#include <FTGLTextureFont.h>
```

Inheritance diagram for FTTextureFont::



3.20.1 Detailed Description

FTTextureFont (p. 64) is a specialisation of the **FTFont** (p. 24) class for handling Texture mapped fonts.

See also:

FTFont (p. 24)

Definition at line 45 of file FTGLTextureFont.h.

Public Member Functions

- **FTTextureFont** (const char ∗fontFilePath)
  Open and read a font file.
- **FTTextureFont** (const unsigned char ∗pBufferBytes, size_t bufferSizeInBytes)
  Open and read a font from a buffer in memory.
- virtual ∼**FTTextureFont** ()
  Destructor.

Protected Member Functions

- virtual **FTGlyph** ∗ **MakeGlyph** (FT_GlyphSlot slot)
  Construct a glyph of the correct type.

3.20.2 Constructor & Destructor Documentation

3.20.2.1 FTTextureFont::FTTextureFont (const char ∗ *fontFilePath*)

Open and read a font file.

Sets Error flag.

Parameters:

fontFilePath font file path.

3.20.2.2 FTTextureFont::FTTextureFont (const unsigned char ∗ *pBufferBytes*, size_t *bufferSizeInBytes*)

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by **FTGL** (p. 3). The pointer must be valid while using **FTGL** (p. 3).

Parameters:

pBufferBytes the in-memory buffer

bufferSizeInBytes the length of the buffer in bytes

3.20.2.3 virtual FTTextureFont::∼FTTextureFont () `[virtual]`

Destructor.

3.20.3 Member Function Documentation

3.20.3.1 virtual FTGlyph∗ FTTextureFont::MakeGlyph (FT_GlyphSlot *slot*) `[protected, virtual]`

Construct a glyph of the correct type.

Clients must override the function and return their specialised **FTGlyph** (p. 35).

Parameters:

slot A FreeType glyph slot.

Returns:

An FT∗∗∗∗Glyph or `null` on failure.

Implements **FTFont** (p. 33).

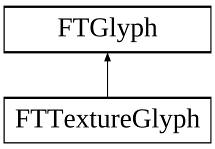
The documentation for this class was generated from the following file:

- **FTGLTextureFont.h**

3.21 FTTextureGlyph Class Reference

`#include <FTTextureGlyph.h>`

Inheritance diagram for FTTextureGlyph::



3.21.1 Detailed Description

FTTextureGlyph (p. 66) is a specialisation of **FTGlyph** (p. 35) for creating texture glyphs.

Definition at line 43 of file FTTextureGlyph.h.

Public Member Functions

- **FTTextureGlyph** (FT_GlyphSlot glyph, int id, int xOffset, int yOffset, int width, int height)
  Constructor.
- virtual ∼**FTTextureGlyph** ()
  Destructor.
- virtual const **FTPoint** & **Render** (const **FTPoint** &pen, int renderMode)
  Render this glyph at the current pen position.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 FTTextureGlyph::FTTextureGlyph (FT_GlyphSlot *glyph*, int *id*, int *xOffset*, int *yOffset*, int *width*, int *height*)

Constructor.

Parameters:

glyph The Freetype glyph to be processed
id The id of the texture that this glyph will be drawn in
xOffset The x offset into the parent texture to draw this glyph
yOffset The y offset into the parent texture to draw this glyph
width The width of the parent texture
height The height (number of rows) of the parent texture

3.21.2.2 virtual FTTextureGlyph::∼FTTextureGlyph () `[virtual]`

Destructor.

3.21.3 Member Function Documentation

3.21.3.1 virtual const FTPoint& FTTextureGlyph::Render (const FTPoint & *pen*, int *renderMode*) `[virtual]`

Render this glyph at the current pen position.

Parameters:

pen The current pen position.

renderMode Render mode to display

Returns:

The advance distance for this glyph.

Implements **FTGlyph** (p. 36).

The documentation for this class was generated from the following file:

- **FTTextureGlyph.h**

Chapter 4

File Documentation

4.1 faq.dox File Reference

4.2 FTBBox.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTBBox**

  FTBBox *(*p. *5) is a convenience class for handling bounding boxes.*

4.3 FTBitmapGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTBitmapGlyph**

  FTBitmapGlyph *(*p. *11) is a specialisation of* ***FTGlyph*** *(*p. *35) for creating bitmaps.*

Functions

- **FTGLglyph** $*$ **ftglCreateBitmapGlyph** (FT_GlyphSlot glyph)

  Create a specialisation of FTGLglyph for creating bitmaps.

4.3.1 Function Documentation

4.3.1.1 FTGLglyph$*$ ftglCreateBitmapGlyph (FT_GlyphSlot *glyph*)

Create a specialisation of FTGLglyph for creating bitmaps.

Parameters:

glyph The Freetype glyph to be processed

Returns:

An FTGLglyph$*$ object.

4.4 FTBuffer.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTBuffer**

  FTBuffer *(*p. *13) is a helper class for pixel buffers.*

4.5 FTBufferFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTBufferFont**

  FTBufferFont *(*p. *16) is a specialisation of the* ***FTFont*** *(*p. *24) class for handling memory buffer fonts.*

Functions

- **FTGLfont** ∗ **ftglCreateBufferFont** (const char ∗file)

  Create a specialised FTGLfont object for handling memory buffer fonts.

4.5.1 Function Documentation

4.5.1.1 FTGLfont∗ ftglCreateBufferFont (const char ∗ *file*)

Create a specialised FTGLfont object for handling memory buffer fonts.

Parameters:

file The font file name.

Returns:

An FTGLfont∗ object.

See also:

FTGLfont (p. 77)

4.6 FTBufferGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTBufferGlyph**

  FTBufferGlyph (p. *18) is a specialisation of* ***FTGlyph*** (p. *35) for memory buffer rendering.*

4.7 FTExtrdGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTExtrudeGlyph**

  FTExtrudeGlyph *(p. 22) is a specialisation of* ***FTGlyph*** *(p. 35) for creating tessellated extruded polygon glyphs.*

Defines

- #define **FTExtrdGlyph FTExtrudeGlyph**

Functions

- **FTGLglyph ∗ ftglCreateExtrudeGlyph** (FT_GlyphSlot glyph, float depth, float frontOutset, float backOutset, int useDisplayList)

  Create a specialisation of FTGLglyph for creating tessellated extruded polygon glyphs.

4.7.1 Define Documentation

4.7.1.1 #define FTExtrdGlyph FTExtrudeGlyph

Definition at line 77 of file FTExtrdGlyph.h.

4.7.2 Function Documentation

4.7.2.1 FTGLglyph∗ ftglCreateExtrudeGlyph (FT_GlyphSlot *glyph*, float *depth*, float *frontOutset*, float *backOutset*, int *useDisplayList*)

Create a specialisation of FTGLglyph for creating tessellated extruded polygon glyphs.

Parameters:

glyph The Freetype glyph to be processed

depth The distance along the z axis to extrude the glyph

frontOutset outset contour size

backOutset outset contour size

useDisplayList Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

Returns:

An FTGLglyph∗ object.

4.8 FTFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTFont**

  FTFont *(*p. *24) is the public interface for the* ***FTGL*** *(*p. *3) library.*

Typedefs

- typedef struct _FTGLfont **FTGLfont**

Functions

- **FTGLfont** ∗ **ftglCreateCustomFont** (char const ∗fontFilePath, void ∗data, **FTGLglyph** ∗(∗makeglyphCallback)(FT_GlyphSlot, void ∗))

  Create a custom ***FTGL*** *(*p. *3) font object.*

- void **ftglDestroyFont** (**FTGLfont** ∗font)

  Destroy an ***FTGL*** *(*p. *3) font object.*

- int **ftglAttachFile** (**FTGLfont** ∗font, const char ∗path)

  Attach auxilliary file to font e.g.

- int **ftglAttachData** (**FTGLfont** ∗font, const unsigned char ∗data, size_t size)

  Attach auxilliary data to font, e.g.

- int **ftglSetFontCharMap** (**FTGLfont** ∗font, FT_Encoding encoding)

  Set the character map for the face.

- unsigned int **ftglGetFontCharMapCount** (**FTGLfont** ∗font)

  Get the number of character maps in this face.

- FT_Encoding ∗ **ftglGetFontCharMapList** (**FTGLfont** ∗font)

  Get a list of character maps in this face.

- int **ftglSetFontFaceSize** (**FTGLfont** ∗font, unsigned int size, unsigned int res)

  Set the char size for the current face.

- unsigned int **ftglGetFontFaceSize** (**FTGLfont** ∗font)

  Get the current face size in points (1/72 inch).

- void **ftglSetFontDepth** (**FTGLfont** ∗font, float depth)

  Set the extrusion distance for the font.

- void **ftglSetFontOutset** (**FTGLfont** ∗font, float front, float back)

Set the outset distance for the font.

- void **ftglSetFontDisplayList** (**FTGLfont** ∗font, int useList)
  *Enable or disable the use of Display Lists inside **FTGL** (p. 3).*
- float **ftglGetFontAscender** (**FTGLfont** ∗font)
  Get the global ascender height for the face.
- float **ftglGetFontDescender** (**FTGLfont** ∗font)
  Gets the global descender height for the face.
- float **ftglGetFontLineHeight** (**FTGLfont** ∗font)
  Gets the line spacing for the font.
- void **ftglGetFontBBox** (**FTGLfont** ∗font, const char ∗string, int len, float bounds[6])
  Get the bounding box for a string.
- float **ftglGetFontAdvance** (**FTGLfont** ∗font, const char ∗string)
  Get the advance width for a string.
- void **ftglRenderFont** (**FTGLfont** ∗font, const char ∗string, int mode)
  Render a string of characters.
- FT_Error **ftglGetFontError** (**FTGLfont** ∗font)
  Query a font for errors.

4.8.1 Typedef Documentation

4.8.1.1 typedef struct _FTGLfont FTGLfont

Definition at line 399 of file FTFont.h.

4.8.2 Function Documentation

4.8.2.1 int ftglAttachData (FTGLfont ∗ *font*, const unsigned char ∗ *data*, size_t *size*)

Attach auxilliary data to font, e.g.

font metrics, from memory.

Note: not all font formats implement this function.

Parameters:

font An FTGLfont∗ object.
data The in-memory buffer.
size The length of the buffer in bytes.

Returns:

1 if file has been attached successfully.

4.8.2.2 int ftglAttachFile (FTGLfont ∗ *font*, const char ∗ *path*)

Attach auxilliary file to font e.g.

font metrics.

Note: not all font formats implement this function.

Parameters:

font An FTGLfont∗ object.

path Auxilliary font file path.

Returns:

1 if file has been attached successfully.

4.8.2.3 FTGLfont∗ ftglCreateCustomFont (char const ∗ *fontFilePath*, void ∗ *data*, FTGLglyph ∗(∗)(FT_GlyphSlot, void ∗) *makeglyphCallback*)

Create a custom **FTGL** (p. 3) font object.

Parameters:

fontFilePath The font file name.

data A pointer to private data that will be passed to callbacks.

makeglyphCallback A glyph-making callback function.

Returns:

An FTGLfont∗ object.

4.8.2.4 void ftglDestroyFont (FTGLfont ∗ *font*)

Destroy an **FTGL** (p. 3) font object.

Parameters:

font An FTGLfont∗ object.

4.8.2.5 float ftglGetFontAdvance (FTGLfont ∗ *font*, const char ∗ *string*)

Get the advance width for a string.

Parameters:

font An FTGLfont∗ object.

string A char string.

Returns:

Advance width

4.8.2.6 float ftglGetFontAscender (FTGLfont ∗ *font*)

Get the global ascender height for the face.

Parameters:

font An FTGLfont∗ object.

Returns:

Ascender height

4.8.2.7 void ftglGetFontBBox (FTGLfont ∗ *font*, const char ∗ *string*, int *len*, float *bounds*[6])

Get the bounding box for a string.

Parameters:

font An FTGLfont∗ object.

string A char buffer

len The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).

bounds An array of 6 float values where the bounding box's lower left near and upper right far 3D coordinates will be stored.

4.8.2.8 unsigned int ftglGetFontCharMapCount (FTGLfont ∗ *font*)

Get the number of character maps in this face.

Parameters:

font An FTGLfont∗ object.

Returns:

character map count.

4.8.2.9 FT_Encoding∗ ftglGetFontCharMapList (FTGLfont ∗ *font*)

Get a list of character maps in this face.

Parameters:

font An FTGLfont∗ object.

Returns:

pointer to the first encoding.

4.8.2.10 float ftglGetFontDescender (FTGLfont ∗ *font*)

Gets the global descender height for the face.

Parameters:

font An FTGLfont∗ object.

Returns:

Descender height

4.8.2.11 FT_Error ftglGetFontError (FTGLfont ∗ *font*)

Query a font for errors.

Parameters:

font An FTGLfont∗ object.

Returns:

The current error code.

4.8.2.12 unsigned int ftglGetFontFaceSize (FTGLfont ∗ *font*)

Get the current face size in points (1/72 inch).

Parameters:

font An FTGLfont∗ object.

Returns:

face size

4.8.2.13 float ftglGetFontLineHeight (FTGLfont ∗ *font*)

Gets the line spacing for the font.

Parameters:

font An FTGLfont∗ object.

Returns:

Line height

4.8.2.14 void ftglRenderFont (FTGLfont ∗ *font*, const char ∗ *string*, int *mode*)

Render a string of characters.

Parameters:

font An FTGLfont∗ object.

string Char string to be output.

mode Render mode to display.

4.8.2.15 int ftglSetFontCharMap (FTGLfont ∗ *font*, FT_Encoding *encoding*)

Set the character map for the face.

Parameters:

font An FTGLfont∗ object.

encoding Freetype enumerate for char map code.

Returns:

1 if charmap was valid and set correctly.

4.8.2.16 void ftglSetFontDepth (FTGLfont ∗ *font*, float *depth*)

Set the extrusion distance for the font.

Only implemented by **FTExtrudeFont** (p. 20).

Parameters:

font An FTGLfont∗ object.

depth The extrusion distance.

4.8.2.17 void ftglSetFontDisplayList (FTGLfont ∗ *font*, int *useList*)

Enable or disable the use of Display Lists inside **FTGL** (p. 3).

Parameters:

font An FTGLfont∗ object.

useList 1 turns ON display lists. 0 turns OFF display lists.

4.8.2.18 int ftglSetFontFaceSize (FTGLfont ∗ *font*, unsigned int *size*, unsigned int *res*)

Set the char size for the current face.

Parameters:

font An FTGLfont∗ object.

size The face size in points (1/72 inch).

res The resolution of the target device, or 0 to use the default value of 72.

Returns:

1 if size was set correctly.

4.8.2.19 void ftglSetFontOutset (FTGLfont ∗ *font*, float *front*, float *back*)

Set the outset distance for the font.

Only **FTOutlineFont** (p. 41), **FTPolygonFont** (p. 56) and **FTExtrudeFont** (p. 20) implement front outset. Only **FTExtrudeFont** (p. 20) implements back outset.

Parameters:

font An FTGLfont∗ object.

front The front outset distance.

back The back outset distance.

4.9 ftgl.dox File Reference

4.10 ftgl.h File Reference

```
#include <ft2build.h>
#include <FT_FREETYPE_H>
#include <FT_GLYPH_H>
#include <FT_OUTLINE_H>
#include <FTGL/FTPoint.h>
#include <FTGL/FTBBox.h>
#include <FTGL/FTBuffer.h>
#include <FTGL/FTGlyph.h>
#include <FTGL/FTBitmapGlyph.h>
#include <FTGL/FTBufferGlyph.h>
#include <FTGL/FTExtrdGlyph.h>
#include <FTGL/FTOutlineGlyph.h>
#include <FTGL/FTPixmapGlyph.h>
#include <FTGL/FTPolyGlyph.h>
#include <FTGL/FTTextureGlyph.h>
#include <FTGL/FTFont.h>
#include <FTGL/FTGLBitmapFont.h>
#include <FTGL/FTBufferFont.h>
#include <FTGL/FTGLExtrdFont.h>
#include <FTGL/FTGLOutlineFont.h>
#include <FTGL/FTGLPixmapFont.h>
#include <FTGL/FTGLPolygonFont.h>
#include <FTGL/FTGLTextureFont.h>
#include <FTGL/FTLayout.h>
#include <FTGL/FTSimpleLayout.h>
```

Namespaces

- namespace **FTGL**

Defines

- #define **FTGL_BEGIN_C_DECLS** extern "C" { namespace FTGL {
- #define **FTGL_END_C_DECLS** } }
- #define **FTGL_EXPORT**

Typedefs

- typedef double **FTGL_DOUBLE**
- typedef float **FTGL_FLOAT**

Enumerations

- enum **FTGL::RenderMode** { **FTGL::RENDER_FRONT** = 0x0001, **FTGL::RENDER_BACK** = 0x0002, **FTGL::RENDER_SIDE** = 0x0004, **FTGL::RENDER_ALL** = 0xffff }
- enum **FTGL::TextAlignment** { **FTGL::ALIGN_LEFT** = 0, **FTGL::ALIGN_CENTER** = 1, **FTGL::ALIGN_RIGHT** = 2, **FTGL::ALIGN_JUSTIFY** = 3 }

4.10.1 Define Documentation

4.10.1.1 #define FTGL_BEGIN_C_DECLS extern "C" { namespace FTGL {

Definition at line 43 of file ftgl.h.

4.10.1.2 #define FTGL_END_C_DECLS } }

Definition at line 44 of file ftgl.h.

4.10.1.3 #define FTGL_EXPORT

Definition at line 107 of file ftgl.h.

4.10.2 Typedef Documentation

4.10.2.1 typedef double FTGL_DOUBLE

Definition at line 38 of file ftgl.h.

4.10.2.2 typedef float FTGL_FLOAT

Definition at line 39 of file ftgl.h.

4.11 FTGLBitmapFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTBitmapFont**

  FTBitmapFont (p. *9) is a specialisation of the* ***FTFont*** (p. *24) class for handling Bitmap fonts.*

Defines

- #define **FTGLBitmapFont FTBitmapFont**

Functions

- **FTGLfont** ∗ **ftglCreateBitmapFont** (const char ∗file)

  Create a specialised FTGLfont object for handling bitmap fonts.

4.11.1 Define Documentation

4.11.1.1 #define FTGLBitmapFont FTBitmapFont

Definition at line 84 of file FTGLBitmapFont.h.

4.11.2 Function Documentation

4.11.2.1 FTGLfont∗ ftglCreateBitmapFont (const char ∗*file*)

Create a specialised FTGLfont object for handling bitmap fonts.

Parameters:

file The font file name.

Returns:

An FTGLfont∗ object.

See also:

FTGLfont (p. 77)

4.12 FTGLExtrdFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTExtrudeFont**

  FTExtrudeFont *(p. 20) is a specialisation of the* ***FTFont*** *(p. 24) class for handling extruded Polygon fonts.*

Defines

- #define **FTGLExtrdFont FTExtrudeFont**

Functions

- **FTGLfont** ∗ **ftglCreateExtrudeFont** (const char ∗file)

  Create a specialised FTGLfont object for handling extruded poygon fonts.

4.12.1 Define Documentation

4.12.1.1 #define FTGLExtrdFont FTExtrudeFont

Definition at line 85 of file FTGLExtrdFont.h.

4.12.2 Function Documentation

4.12.2.1 FTGLfont∗ ftglCreateExtrudeFont (const char ∗ *file*)

Create a specialised FTGLfont object for handling extruded poygon fonts.

Parameters:

file The font file name.

Returns:

An FTGLfont∗ object.

See also:

FTGLfont (p. 77)
ftglCreatePolygonFont (p. 90)

4.13 FTGLOutlineFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTOutlineFont**

  FTOutlineFont (p. *41) is a specialisation of the* ***FTFont*** (p. *24) class for handling Vector Outline fonts.*

Defines

- #define **FTGLOutlineFont FTOutlineFont**

Functions

- **FTGLfont** ∗ **ftglCreateOutlineFont** (const char ∗file)

  Create a specialised FTGLfont object for handling vector outline fonts.

4.13.1 Define Documentation

4.13.1.1 #define FTGLOutlineFont FTOutlineFont

Definition at line 84 of file FTGLOutlineFont.h.

4.13.2 Function Documentation

4.13.2.1 FTGLfont∗ ftglCreateOutlineFont (const char ∗ *file*)

Create a specialised FTGLfont object for handling vector outline fonts.

Parameters:

file The font file name.

Returns:

An FTGLfont∗ object.

See also:

FTGLfont (p. 77)

4.14 FTGLPixmapFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTPixmapFont**

  FTPixmapFont *(p. 45) is a specialisation of the* ***FTFont*** *(p. 24) class for handling Pixmap (Grey Scale) fonts.*

Defines

- #define **FTGLPixmapFont FTPixmapFont**

Functions

- **FTGLfont** ∗ **ftglCreatePixmapFont** (const char ∗file)

  Create a specialised FTGLfont object for handling pixmap (grey scale) fonts.

4.14.1 Define Documentation

4.14.1.1 #define FTGLPixmapFont FTPixmapFont

Definition at line 84 of file FTGLPixmapFont.h.

4.14.2 Function Documentation

4.14.2.1 FTGLfont∗ ftglCreatePixmapFont (const char ∗ *file*)

Create a specialised FTGLfont object for handling pixmap (grey scale) fonts.

Parameters:

file The font file name.

Returns:

An FTGLfont∗ object.

See also:

FTGLfont (p. 77)

4.15 FTGLPolygonFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTPolygonFont**

  FTPolygonFont *(p. 56) is a specialisation of the* ***FTFont*** *(p. 24) class for handling tesselated Polygon Mesh fonts.*

Defines

- #define **FTGLPolygonFont FTPolygonFont**

Functions

- **FTGLfont** ∗ **ftglCreatePolygonFont** (const char ∗file)

  Create a specialised FTGLfont object for handling tesselated polygon mesh fonts.

4.15.1 Define Documentation

4.15.1.1 #define FTGLPolygonFont FTPolygonFont

Definition at line 84 of file FTGLPolygonFont.h.

4.15.2 Function Documentation

4.15.2.1 FTGLfont∗ ftglCreatePolygonFont (const char ∗ *file*)

Create a specialised FTGLfont object for handling tesselated polygon mesh fonts.

Parameters:

file The font file name.

Returns:

An FTGLfont∗ object.

See also:

FTGLfont (p. 77)

4.16 FTGLTextureFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTTextureFont**

  FTTextureFont *(*p. *64) is a specialisation of the* ***FTFont*** *(*p. *24) class for handling Texture mapped fonts.*

Defines

- #define **FTGLTextureFont FTTextureFont**

Functions

- **FTGLfont** ∗ **ftglCreateTextureFont** (const char ∗file)

  Create a specialised FTGLfont object for handling texture-mapped fonts.

4.16.1 Define Documentation

4.16.1.1 #define FTGLTextureFont FTTextureFont

Definition at line 84 of file FTGLTextureFont.h.

4.16.2 Function Documentation

4.16.2.1 FTGLfont∗ ftglCreateTextureFont (const char ∗ *file*)

Create a specialised FTGLfont object for handling texture-mapped fonts.

Parameters:

file The font file name.

Returns:

An FTGLfont∗ object.

See also:

FTGLfont (p. 77)

4.17 FTGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTGlyph**

  FTGlyph *(*p. *35) is the base class for* ***FTGL*** *(*p. *3) glyphs.*

Typedefs

- typedef struct _FTGLglyph **FTGLglyph**

Functions

- **FTGLglyph** ∗ **ftglCreateCustomGlyph** (**FTGLglyph** ∗base, void ∗data, void(∗renderCallback)(**FTGLglyph** ∗, void ∗, **FTGL_DOUBLE**, **FTGL_DOUBLE**, int, **FTGL_-DOUBLE** ∗, **FTGL_DOUBLE** ∗), void(∗destroyCallback)(**FTGLglyph** ∗, void ∗))

  Create a custom ***FTGL*** *(*p. *3) glyph object.*

- void **ftglDestroyGlyph** (**FTGLglyph** ∗glyph)

  Destroy an ***FTGL*** *(*p. *3) glyph object.*

- void **ftglRenderGlyph** (**FTGLglyph** ∗glyph, **FTGL_DOUBLE** penx, **FTGL_DOUBLE** peny, int renderMode, **FTGL_DOUBLE** ∗advancex, **FTGL_DOUBLE** ∗advancey)

  Render a glyph at the current pen position and compute the corresponding advance.

- float **ftglGetGlyphAdvance** (**FTGLglyph** ∗glyph)

  Return the advance for a glyph.

- void **ftglGetGlyphBBox** (**FTGLglyph** ∗glyph, float bounds[6])

  Return the bounding box for a glyph.

- FT_Error **ftglGetGlyphError** (**FTGLglyph** ∗glyph)

  Query a glyph for errors.

4.17.1 Typedef Documentation

4.17.1.1 typedef struct _FTGLglyph FTGLglyph

Definition at line 133 of file FTGlyph.h.

4.17.2 Function Documentation

4.17.2.1 FTGLglyph∗ ftglCreateCustomGlyph (FTGLglyph ∗ *base*, void ∗ *data*, void(∗)(FTGLglyph ∗, void ∗, FTGL_DOUBLE, FTGL_DOUBLE, int, FTGL_DOUBLE ∗, FTGL_DOUBLE ∗) *renderCallback*, void(∗)(FTGLglyph ∗, void ∗) *destroyCallback*)

Create a custom **FTGL** (p. 3) glyph object.

FIXME: maybe get rid of "base" and have advanceCallback etc. functions

Parameters:

base The base FTGLglyph∗ to subclass.

data A pointer to private data that will be passed to callbacks.

renderCallback A rendering callback function.

destroyCallback A callback function to be called upon destruction.

Returns:

An FTGLglyph∗ object.

4.17.2.2 void ftglDestroyGlyph (FTGLglyph ∗ *glyph*)

Destroy an **FTGL** (p. 3) glyph object.

Parameters:

glyph An FTGLglyph∗ object.

4.17.2.3 float ftglGetGlyphAdvance (FTGLglyph ∗ *glyph*)

Return the advance for a glyph.

Parameters:

glyph An FTGLglyph∗ object.

Returns:

The advance's X component.

4.17.2.4 void ftglGetGlyphBBox (FTGLglyph ∗ *glyph*, float *bounds*[6])

Return the bounding box for a glyph.

Parameters:

glyph An FTGLglyph∗ object.

bounds An array of 6 float values where the bounding box's lower left near and upper right far 3D coordinates will be stored.

4.17.2.5 FT_Error ftglGetGlyphError (FTGLglyph ∗ *glyph*)

Query a glyph for errors.

Parameters:

glyph An FTGLglyph∗ object.

Returns:

The current error code.

4.17.2.6 void ftglRenderGlyph (FTGLglyph ∗ *glyph*, FTGL_DOUBLE *penx*, FTGL_DOUBLE *peny*, int *renderMode*, FTGL_DOUBLE ∗ *advancex*, FTGL_DOUBLE ∗ *advancey*)

Render a glyph at the current pen position and compute the corresponding advance.

Parameters:

glyph An FTGLglyph∗ object.

penx The current pen's X position.

peny The current pen's Y position.

renderMode Render mode to display

advancex A pointer to an FTGL_DOUBLE where to write the advance's X component.

advancey A pointer to an FTGL_DOUBLE where to write the advance's Y component.

4.18 FTLayout.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTLayout**

  FTLayout *(*p. *38) is the interface for layout managers that render text.*

Typedefs

- typedef struct _FTGLlayout **FTGLlayout**

Functions

- void **ftglDestroyLayout** (**FTGLlayout** ∗layout)

  Destroy an ***FTGL*** *(*p. *3) layout object.*

- void **ftglGetLayoutBBox** (**FTGLlayout** ∗layout, const char ∗string, float bounds[6])

  Get the bounding box for a string.

- void **ftglRenderLayout** (**FTGLlayout** ∗layout, const char ∗string, int mode)

  Render a string of characters.

- FT_Error **ftglGetLayoutError** (**FTGLlayout** ∗layout)

  Query a layout for errors.

4.18.1 Typedef Documentation

4.18.1.1 typedef struct _FTGLlayout FTGLlayout

Definition at line 151 of file FTLayout.h.

4.18.2 Function Documentation

4.18.2.1 void ftglDestroyLayout (FTGLlayout ∗ *layout*)

Destroy an **FTGL** (p. 3) layout object.

Parameters:

layout An FTGLlayout∗ object.

4.18.2.2 void ftglGetLayoutBBox (FTGLlayout ∗ *layout*, const char ∗ *string*, float *bounds*[6])

Get the bounding box for a string.

Parameters:

layout An FTGLlayout∗ object.

string A char buffer

bounds An array of 6 float values where the bounding box's lower left near and upper right far 3D coordinates will be stored.

4.18.2.3 FT_Error ftglGetLayoutError (FTGLlayout ∗ *layout*)

Query a layout for errors.

Parameters:

layout An FTGLlayout∗ object.

Returns:

The current error code.

4.18.2.4 void ftglRenderLayout (FTGLlayout ∗ *layout*, const char ∗ *string*, int *mode*)

Render a string of characters.

Parameters:

layout An FTGLlayout∗ object.

string Char string to be output.

mode Render mode to display.

4.19 FTOutlineGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTOutlineGlyph**

  ***FTOutlineGlyph** (p. 43) is a specialisation of **FTGlyph** (p. 35) for creating outlines.*

Functions

- **FTGLglyph ∗ ftglCreateOutlineGlyph** (FT_GlyphSlot glyph, float outset, int useDisplayList)

  Create a specialisation of FTGLglyph for creating outlines.

4.19.1 Function Documentation

4.19.1.1 FTGLglyph∗ ftglCreateOutlineGlyph (FT_GlyphSlot *glyph*, float *outset*, int *useDisplayList*)

Create a specialisation of FTGLglyph for creating outlines.

Parameters:

glyph The Freetype glyph to be processed

outset outset contour size

useDisplayList Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

Returns:

An FTGLglyph∗ object.

4.20 FTPixmapGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTPixmapGlyph**

  FTPixmapGlyph (p. *47) is a specialisation of* ***FTGlyph*** (p. *35) for creating pixmaps.*

Functions

- **FTGLglyph** ∗ **ftglCreatePixmapGlyph** (FT_GlyphSlot glyph)

  Create a specialisation of FTGLglyph for creating pixmaps.

4.20.1 Function Documentation

4.20.1.1 FTGLglyph∗ ftglCreatePixmapGlyph (FT_GlyphSlot *glyph*)

Create a specialisation of FTGLglyph for creating pixmaps.

Parameters:

glyph The Freetype glyph to be processed

Returns:

An FTGLglyph∗ object.

4.21 FTPoint.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTPoint**

  ***FTPoint** (p. 49) class is a basic 3-dimensional point or vector.*

4.22 FTPolyGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTPolygonGlyph**

  FTPolygonGlyph (p. *58) is a specialisation of* ***FTGlyph*** (p. *35) for creating tessellated polygon glyphs.*

Defines

- #define **FTPolyGlyph FTPolygonGlyph**

Functions

- **FTGLglyph** $*$ **ftglCreatePolygonGlyph** (FT_GlyphSlot glyph, float outset, int useDisplayList)

  Create a specialisation of FTGLglyph for creating tessellated polygon glyphs.

4.22.1 Define Documentation

4.22.1.1 #define FTPolyGlyph FTPolygonGlyph

Definition at line 74 of file FTPolyGlyph.h.

4.22.2 Function Documentation

4.22.2.1 FTGLglyph$*$ ftglCreatePolygonGlyph (FT_GlyphSlot *glyph*, float *outset*, int *useDisplayList*)

Create a specialisation of FTGLglyph for creating tessellated polygon glyphs.

Parameters:

glyph The Freetype glyph to be processed

outset outset contour size

useDisplayList Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

Returns:

An FTGLglyph$*$ object.

4.23 FTSimpleLayout.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTSimpleLayout**

  FTSimpleLayout *(p. 60) is a specialisation of* ***FTLayout*** *(p. 38) for simple text boxes.*

Functions

- **FTGLlayout** ∗ **ftglCreateSimpleLayout** (void)
- void **ftglSetLayoutFont** (**FTGLlayout** ∗, **FTGLfont** ∗)
- **FTGLfont** ∗ **ftglGetLayoutFont** (**FTGLlayout** ∗)
- void **ftglSetLayoutLineLength** (**FTGLlayout** ∗, const float)
- float **ftglGetLayoutLineLength** (**FTGLlayout** ∗)
- void **ftglSetLayoutAlignment** (**FTGLlayout** ∗, const int)
- int **ftglGetLayoutAlignement** (**FTGLlayout** ∗)
- void **ftglSetLayoutLineSpacing** (**FTGLlayout** ∗, const float)
- float **ftglGetLayoutLineSpacing** (**FTGLlayout** ∗)

4.23.1 Function Documentation

4.23.1.1 FTGLlayout∗ ftglCreateSimpleLayout (void)

4.23.1.2 int ftglGetLayoutAlignement (FTGLlayout ∗)

4.23.1.3 FTGLfont∗ ftglGetLayoutFont (FTGLlayout ∗)

4.23.1.4 float ftglGetLayoutLineLength (FTGLlayout ∗)

4.23.1.5 float ftglGetLayoutLineSpacing (FTGLlayout ∗)

4.23.1.6 void ftglSetLayoutAlignment (FTGLlayout ∗, const *int*)

4.23.1.7 void ftglSetLayoutFont (FTGLlayout ∗, FTGLfont ∗)

4.23.1.8 void ftglSetLayoutLineLength (FTGLlayout ∗, const *float*)

4.23.1.9 void ftglSetLayoutLineSpacing (FTGLlayout ∗, const *float*)

4.24 FTTextureGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class **FTTextureGlyph**

  FTTextureGlyph *(p. 66) is a specialisation of* ***FTGlyph*** *(p. 35) for creating texture glyphs.*

Functions

- **FTGLglyph** ∗ **ftglCreateTextureGlyph** (FT_GlyphSlot glyph, int id, int xOffset, int yOffset, int width, int height)

  Create a specialisation of FTGLglyph for creating pixmaps.

4.24.1 Function Documentation

4.24.1.1 FTGLglyph∗ ftglCreateTextureGlyph (FT_GlyphSlot *glyph*, int *id*, int *xOffset*, int *yOffset*, int *width*, int *height*)

Create a specialisation of FTGLglyph for creating pixmaps.

Parameters:

glyph The Freetype glyph to be processed.

id The id of the texture that this glyph will be drawn in.

xOffset The x offset into the parent texture to draw this glyph.

yOffset The y offset into the parent texture to draw this glyph.

width The width of the parent texture.

height The height (number of rows) of the parent texture.

Returns:

An FTGLglyph∗ object.

4.25 projects_using_ftgl.txt File Reference

4.26 tutorial.dox File Reference

Index