

FTGL

2.1.3~rc5

Generated by Doxygen 1.8.8

Mon Apr 27 2020 09:45:59

Contents

1 FTGL User Guide	1
1.1 Introduction	1
1.2 Documentation	1
1.3 Additional information	2
2 Frequently Asked Questions	3
2.1 FAQ	3
2.1.1 When I try to compile %FTGL it complains about a missing file from the include: #include <ft2build.h>	3
2.1.2 Is it possible to map a font to a "unit" size? My application relies on the fonts being a certain "physical" height (in OpenGL coordinate space) rather than a point size in display space. Any thoughts/suggestions?	3
3 Projects using FTGL	5
3.1 %FTGL language bindings	5
3.1.1 %FTGL#	5
3.1.2 GIGuiA	5
3.1.3 Ruby %FTGL	5
3.1.4 PyFTGL	5
3.2 Projects currently using %FTGL	5
3.2.1 Agent World	5
3.2.2 Amaltheia	6
3.2.3 Armagetron Advanced	6
3.2.4 Audicle	6
3.2.5 Battlestar T.U.X.	6
3.2.6 BJS	6
3.2.7 Blender	6
3.2.8 Breve	6
3.2.9 BZFlag	6
3.2.10 Capture The Flag	7
3.2.11 Cello	7
3.2.12 Chimera	7
3.2.13 Cinepaint	7

3.2.14	Duel	7
3.2.15	Empty Clip	7
3.2.16	Freebox	7
3.2.17	Gem	7
3.2.18	GLMayan	7
3.2.19	Glover	8
3.2.20	Ivf++	8
3.2.21	Jahshaka	8
3.2.22	Karaoke FX	8
3.2.23	Libinstrudeo	8
3.2.24	Light Speed!	8
3.2.25	MySQL GUI Tools	8
3.2.26	OctPlot	8
3.2.27	Open ActiveWrl	9
3.2.28	OpenEagles	9
3.2.29	OpenGC	9
3.2.30	OpenSG	9
3.2.31	Panthera	9
3.2.32	Planet Penguin Racer	9
3.2.33	projectM	9
3.2.34	Puzzle Bobble 3D	9
3.2.35	ROOT	10
3.2.36	SCIRun	10
3.2.37	TINE	10
3.2.38	Tiny Planet	10
3.2.39	Truevision	10
3.2.40	Tulip	10
3.2.41	Ubit	10
3.2.42	VRS	10
3.2.43	VTK	10
3.2.44	XLock	11
3.3	Projects that used to use %FTGL	11
3.3.1	GNU Backgammon	11
3.3.2	OpenSceneGraph	11
3.3.3	Teddy	11
3.3.4	VigiPac	11
4	FTGL tutorial	13
4.1	Starting to use %FTGL	13
4.2	Choosing a font type	13

4.2.1	Raster fonts	13
4.2.2	Vector fonts	13
4.2.3	Textured fonts	14
4.3	Create font objects	14
4.3.1	in C	14
4.3.2	in C++	14
4.4	More font commands	15
4.4.1	Font metrics	15
4.4.2	Specifying a character map encoding	15
4.5	Sample font manager class	16
5	Namespace Documentation	19
5.1	FTGL Namespace Reference	19
5.1.1	Enumeration Type Documentation	19
5.1.1.1	RenderMode	19
5.1.1.2	TextAlignment	19
6	Data Structure Documentation	21
6.1	FTBBox Class Reference	21
6.1.1	Detailed Description	21
6.1.2	Constructor & Destructor Documentation	22
6.1.2.1	FTBBox	22
6.1.2.2	FTBBox	22
6.1.2.3	FTBBox	22
6.1.2.4	FTBBox	22
6.1.2.5	~FTBBox	22
6.1.3	Member Function Documentation	22
6.1.3.1	Invalidate	22
6.1.3.2	IsValid	22
6.1.3.3	Lower	23
6.1.3.4	operator+=	23
6.1.3.5	operator" =	23
6.1.3.6	SetDepth	23
6.1.3.7	Upper	23
6.2	FTBitmapFont Class Reference	23
6.2.1	Detailed Description	24
6.2.2	Constructor & Destructor Documentation	24
6.2.2.1	FTBitmapFont	24
6.2.2.2	FTBitmapFont	24
6.2.2.3	~FTBitmapFont	24
6.2.3	Member Function Documentation	25

6.2.3.1	MakeGlyph	25
6.3	FTBitmapGlyph Class Reference	25
6.3.1	Detailed Description	25
6.3.2	Constructor & Destructor Documentation	26
6.3.2.1	FTBitmapGlyph	26
6.3.2.2	~FTBitmapGlyph	27
6.3.3	Member Function Documentation	27
6.3.3.1	Render	27
6.4	FTBuffer Class Reference	27
6.4.1	Detailed Description	28
6.4.2	Constructor & Destructor Documentation	28
6.4.2.1	FTBuffer	28
6.4.2.2	~FTBuffer	28
6.4.3	Member Function Documentation	28
6.4.3.1	Height	28
6.4.3.2	Pixels	28
6.4.3.3	Pos	28
6.4.3.4	Pos	29
6.4.3.5	Size	30
6.4.3.6	Width	30
6.5	FTBufferFont Class Reference	30
6.5.1	Detailed Description	31
6.5.2	Constructor & Destructor Documentation	31
6.5.2.1	FTBufferFont	31
6.5.2.2	FTBufferFont	31
6.5.2.3	~FTBufferFont	31
6.5.3	Member Function Documentation	31
6.5.3.1	MakeGlyph	31
6.6	FTBufferGlyph Class Reference	32
6.6.1	Detailed Description	32
6.6.2	Constructor & Destructor Documentation	32
6.6.2.1	FTBufferGlyph	32
6.6.2.2	~FTBufferGlyph	32
6.6.3	Member Function Documentation	32
6.6.3.1	Render	32
6.7	FTExtrudeFont Class Reference	33
6.7.1	Detailed Description	33
6.7.2	Constructor & Destructor Documentation	34
6.7.2.1	FTExtrudeFont	34
6.7.2.2	FTExtrudeFont	34

6.7.2.3	~FTExtrudeFont	34
6.7.3	Member Function Documentation	34
6.7.3.1	MakeGlyph	34
6.8	FTExtrudeGlyph Class Reference	34
6.8.1	Detailed Description	35
6.8.2	Constructor & Destructor Documentation	35
6.8.2.1	FTExtrudeGlyph	35
6.8.2.2	~FTExtrudeGlyph	35
6.8.3	Member Function Documentation	35
6.8.3.1	Render	35
6.9	FTFont Class Reference	36
6.9.1	Detailed Description	38
6.9.2	Constructor & Destructor Documentation	38
6.9.2.1	FTFont	38
6.9.2.2	FTFont	38
6.9.2.3	~FTFont	38
6.9.3	Member Function Documentation	38
6.9.3.1	Advance	38
6.9.3.2	Advance	39
6.9.3.3	Ascender	39
6.9.3.4	Attach	39
6.9.3.5	Attach	39
6.9.3.6	BBox	40
6.9.3.7	BBox	40
6.9.3.8	BBox	40
6.9.3.9	BBox	41
6.9.3.10	CharMap	41
6.9.3.11	CharMapCount	41
6.9.3.12	CharMapList	41
6.9.3.13	Depth	42
6.9.3.14	Descender	42
6.9.3.15	Error	42
6.9.3.16	FaceSize	42
6.9.3.17	FaceSize	42
6.9.3.18	GlyphLoadFlags	42
6.9.3.19	LineHeight	43
6.9.3.20	MakeGlyph	43
6.9.3.21	Outset	43
6.9.3.22	Outset	43
6.9.3.23	Render	43

6.9.3.24	Render	44
6.9.3.25	UseDisplayList	44
6.9.4	Friends And Related Function Documentation	44
6.9.4.1	FTBitmapFont	44
6.9.4.2	FTBufferFont	44
6.9.4.3	FTExtrudeFont	44
6.9.4.4	FTFontImpl	45
6.9.4.5	FTOutlineFont	45
6.9.4.6	FTPixmapFont	45
6.9.4.7	FTPolygonFont	45
6.9.4.8	FTTextureFont	45
6.10	FTGlyph Class Reference	45
6.10.1	Detailed Description	46
6.10.2	Constructor & Destructor Documentation	46
6.10.2.1	FTGlyph	46
6.10.2.2	~FTGlyph	46
6.10.3	Member Function Documentation	46
6.10.3.1	Advance	46
6.10.3.2	BBox	47
6.10.3.3	Error	47
6.10.3.4	Render	47
6.10.4	Friends And Related Function Documentation	47
6.10.4.1	FTBitmapGlyph	47
6.10.4.2	FTBufferGlyph	47
6.10.4.3	FTExtrudeGlyph	47
6.10.4.4	FTOutlineGlyph	47
6.10.4.5	FTPixmapGlyph	47
6.10.4.6	FTPolygonGlyph	48
6.10.4.7	FTTextureGlyph	48
6.11	FTLayout Class Reference	48
6.11.1	Detailed Description	49
6.11.2	Constructor & Destructor Documentation	49
6.11.2.1	FTLayout	49
6.11.2.2	~FTLayout	49
6.11.3	Member Function Documentation	49
6.11.3.1	BBox	49
6.11.3.2	BBox	49
6.11.3.3	Error	50
6.11.3.4	Render	50
6.11.3.5	Render	50

6.11.4	Friends And Related Function Documentation	50
6.11.4.1	FTSimpleLayout	50
6.12	FTOutlineFont Class Reference	51
6.12.1	Detailed Description	51
6.12.2	Constructor & Destructor Documentation	51
6.12.2.1	FTOutlineFont	51
6.12.2.2	FTOutlineFont	51
6.12.2.3	~FTOutlineFont	52
6.12.3	Member Function Documentation	52
6.12.3.1	MakeGlyph	52
6.13	FTOutlineGlyph Class Reference	52
6.13.1	Detailed Description	53
6.13.2	Constructor & Destructor Documentation	53
6.13.2.1	FTOutlineGlyph	53
6.13.2.2	~FTOutlineGlyph	53
6.13.3	Member Function Documentation	53
6.13.3.1	Render	53
6.14	FTPixmapFont Class Reference	53
6.14.1	Detailed Description	54
6.14.2	Constructor & Destructor Documentation	54
6.14.2.1	FTPixmapFont	54
6.14.2.2	FTPixmapFont	54
6.14.2.3	~FTPixmapFont	55
6.14.3	Member Function Documentation	55
6.14.3.1	MakeGlyph	55
6.15	FTPixmapGlyph Class Reference	55
6.15.1	Detailed Description	56
6.15.2	Constructor & Destructor Documentation	56
6.15.2.1	FTPixmapGlyph	56
6.15.2.2	~FTPixmapGlyph	56
6.15.3	Member Function Documentation	56
6.15.3.1	Render	56
6.16	FTPoint Class Reference	56
6.16.1	Detailed Description	57
6.16.2	Constructor & Destructor Documentation	58
6.16.2.1	FTPoint	58
6.16.2.2	FTPoint	58
6.16.2.3	FTPoint	58
6.16.3	Member Function Documentation	58
6.16.3.1	Normalise	58

6.16.3.2	operator const FTGL_DOUBLE *	58
6.16.3.3	operator*	58
6.16.3.4	operator+	59
6.16.3.5	operator+=	59
6.16.3.6	operator-	59
6.16.3.7	operator-=	59
6.16.3.8	operator^	60
6.16.3.9	X	60
6.16.3.10	X	60
6.16.3.11	Xf	60
6.16.3.12	Y	60
6.16.3.13	Y	60
6.16.3.14	Yf	60
6.16.3.15	Z	61
6.16.3.16	Z	61
6.16.3.17	Zf	61
6.16.4	Friends And Related Function Documentation	61
6.16.4.1	operator"!=	61
6.16.4.2	operator*	61
6.16.4.3	operator*	61
6.16.4.4	operator==	62
6.17	FTPolygonFont Class Reference	62
6.17.1	Detailed Description	63
6.17.2	Constructor & Destructor Documentation	63
6.17.2.1	FTPolygonFont	63
6.17.2.2	FTPolygonFont	63
6.17.2.3	~FTPolygonFont	63
6.17.3	Member Function Documentation	63
6.17.3.1	MakeGlyph	63
6.18	FTPolygonGlyph Class Reference	64
6.18.1	Detailed Description	64
6.18.2	Constructor & Destructor Documentation	64
6.18.2.1	FTPolygonGlyph	64
6.18.2.2	~FTPolygonGlyph	64
6.18.3	Member Function Documentation	64
6.18.3.1	Render	65
6.19	FTSimpleLayout Class Reference	66
6.19.1	Detailed Description	67
6.19.2	Constructor & Destructor Documentation	67
6.19.2.1	FTSimpleLayout	67

6.19.2.2	~FTSimpleLayout	67
6.19.3	Member Function Documentation	67
6.19.3.1	BBox	67
6.19.3.2	BBox	67
6.19.3.3	GetAlignment	68
6.19.3.4	GetFont	68
6.19.3.5	GetLineLength	68
6.19.3.6	GetLineSpacing	68
6.19.3.7	Render	68
6.19.3.8	Render	68
6.19.3.9	SetAlignment	69
6.19.3.10	SetFont	69
6.19.3.11	SetLineLength	69
6.19.3.12	SetLineSpacing	69
6.20	FTTextureFont Class Reference	69
6.20.1	Detailed Description	70
6.20.2	Constructor & Destructor Documentation	70
6.20.2.1	FTTextureFont	70
6.20.2.2	FTTextureFont	70
6.20.2.3	~FTTextureFont	71
6.20.3	Member Function Documentation	71
6.20.3.1	MakeGlyph	71
6.21	FTTextureGlyph Class Reference	71
6.21.1	Detailed Description	72
6.21.2	Constructor & Destructor Documentation	72
6.21.2.1	FTTextureGlyph	72
6.21.2.2	~FTTextureGlyph	72
6.21.3	Member Function Documentation	72
6.21.3.1	Render	72
7	File Documentation	73
7.1	faq.dox File Reference	73
7.2	FTBBox.h File Reference	73
7.3	FTBitmapGlyph.h File Reference	73
7.3.1	Function Documentation	73
7.3.1.1	ftglCreateBitmapGlyph	73
7.4	FTBuffer.h File Reference	74
7.5	FTBufferFont.h File Reference	74
7.5.1	Function Documentation	74
7.5.1.1	ftglCreateBufferFont	74

7.6	FTBufferGlyph.h File Reference	75
7.7	FTEextrdGlyph.h File Reference	75
7.7.1	Macro Definition Documentation	75
7.7.1.1	FTEextrdGlyph	75
7.7.2	Function Documentation	75
7.7.2.1	ftglCreateExtrudeGlyph	75
7.8	FTFont.h File Reference	76
7.8.1	Typedef Documentation	77
7.8.1.1	FTGLfont	77
7.8.2	Function Documentation	77
7.8.2.1	ftglAttachData	77
7.8.2.2	ftglAttachFile	77
7.8.2.3	ftglCreateCustomFont	78
7.8.2.4	ftglDestroyFont	78
7.8.2.5	ftglGetFontAdvance	78
7.8.2.6	ftglGetFontAscender	78
7.8.2.7	ftglGetFontBBox	79
7.8.2.8	ftglGetFontCharMapCount	80
7.8.2.9	ftglGetFontCharMapList	80
7.8.2.10	ftglGetFontDescender	80
7.8.2.11	ftglGetFontError	80
7.8.2.12	ftglGetFontFaceSize	81
7.8.2.13	ftglGetFontLineHeight	82
7.8.2.14	ftglRenderFont	82
7.8.2.15	ftglSetFontCharMap	82
7.8.2.16	ftglSetFontDepth	82
7.8.2.17	ftglSetFontDisplayList	83
7.8.2.18	ftglSetFontFaceSize	83
7.8.2.19	ftglSetFontOutset	83
7.9	ftgl.dox File Reference	83
7.10	ftgl.h File Reference	83
7.10.1	Macro Definition Documentation	84
7.10.1.1	FTGL_BEGIN_C_DECLS	84
7.10.1.2	FTGL_END_C_DECLS	85
7.10.1.3	FTGL_EXPORT	85
7.10.2	Typedef Documentation	85
7.10.2.1	FTGL_DOUBLE	85
7.10.2.2	FTGL_FLOAT	85
7.11	FTGLBitmapFont.h File Reference	85
7.11.1	Macro Definition Documentation	85

7.11.1.1	FTGLBitmapFont	85
7.11.2	Function Documentation	85
7.11.2.1	ftglCreateBitmapFont	85
7.12	FTGLExtrdFont.h File Reference	86
7.12.1	Macro Definition Documentation	86
7.12.1.1	FTGLExtrdFont	86
7.12.2	Function Documentation	86
7.12.2.1	ftglCreateExtrudeFont	86
7.13	FTGLOutlineFont.h File Reference	87
7.13.1	Macro Definition Documentation	87
7.13.1.1	FTGLOutlineFont	87
7.13.2	Function Documentation	87
7.13.2.1	ftglCreateOutlineFont	87
7.14	FTGLPixmapFont.h File Reference	87
7.14.1	Macro Definition Documentation	88
7.14.1.1	FTGLPixmapFont	88
7.14.2	Function Documentation	88
7.14.2.1	ftglCreatePixmapFont	88
7.15	FTGLPolygonFont.h File Reference	88
7.15.1	Macro Definition Documentation	89
7.15.1.1	FTGLPolygonFont	89
7.15.2	Function Documentation	89
7.15.2.1	ftglCreatePolygonFont	89
7.16	FTGLTextureFont.h File Reference	89
7.16.1	Macro Definition Documentation	89
7.16.1.1	FTGLTextureFont	89
7.16.2	Function Documentation	90
7.16.2.1	ftglCreateTextureFont	90
7.17	FTGlyph.h File Reference	91
7.17.1	Typedef Documentation	91
7.17.1.1	FTGLglyph	91
7.17.2	Function Documentation	92
7.17.2.1	ftglCreateCustomGlyph	92
7.17.2.2	ftglDestroyGlyph	92
7.17.2.3	ftglGetGlyphAdvance	92
7.17.2.4	ftglGetGlyphBBox	92
7.17.2.5	ftglGetGlyphError	92
7.17.2.6	ftglRenderGlyph	93
7.18	FTLayout.h File Reference	93
7.18.1	Typedef Documentation	93

7.18.1.1	FTGLLayout	93
7.18.2	Function Documentation	94
7.18.2.1	ftglDestroyLayout	94
7.18.2.2	ftglGetLayoutBBox	94
7.18.2.3	ftglGetLayoutError	94
7.18.2.4	ftglRenderLayout	94
7.19	FTOutlineGlyph.h File Reference	94
7.19.1	Function Documentation	95
7.19.1.1	ftglCreateOutlineGlyph	95
7.20	FTPixmapGlyph.h File Reference	95
7.20.1	Function Documentation	95
7.20.1.1	ftglCreatePixmapGlyph	95
7.21	FTPoint.h File Reference	96
7.22	FTPolyGlyph.h File Reference	96
7.22.1	Macro Definition Documentation	96
7.22.1.1	FTPolyGlyph	96
7.22.2	Function Documentation	96
7.22.2.1	ftglCreatePolygonGlyph	96
7.23	FTSimpleLayout.h File Reference	97
7.23.1	Function Documentation	97
7.23.1.1	ftglCreateSimpleLayout	97
7.23.1.2	ftglGetLayoutAlignement	97
7.23.1.3	ftglGetLayoutFont	97
7.23.1.4	ftglGetLayoutLineLength	97
7.23.1.5	ftglGetLayoutLineSpacing	97
7.23.1.6	ftglSetLayoutAlignment	97
7.23.1.7	ftglSetLayoutFont	97
7.23.1.8	ftglSetLayoutLineLength	97
7.23.1.9	ftglSetLayoutLineSpacing	97
7.24	FTTextureGlyph.h File Reference	97
7.24.1	Function Documentation	98
7.24.1.1	ftglCreateTextureGlyph	98
7.25	projects_using_ftgl.txt File Reference	98
7.26	tutorial.dox File Reference	98
Index		99

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. 13)
- C API reference:
 - **FTGlyph.h** (p. 91)
 - **FTFont.h** (p. 76)
 - **FTLayout.h** (p. 93)
- C++ API reference:

- class **FTGlyph** (p. 45)
- class **FTFont** (p. 36)
- class **FTLayout** (p. 48)

1.3 Additional information

- **Frequently Asked Questions** (p. 3)
- **Projects using FTGL** (p. 5)

Chapter 2

Frequently Asked Questions

2.1 FAQ

2.1.1 When I try to compile %FTGL it complains about a missing file from the include: #include <ft2build.h>

FTGL relies on FreeType 2 for opening and decoding font files. This include is the main include for FreeType. You will need to download Freetype 2 and install it. Then make sure that the FTGL project that you are using points to your FreeType installation.

2.1.2 Is it possible to map a font to a "unit" size? My application relies on the fonts being a certain "physical" height (in OpenGL coordinate space) rather than a point size in display space. Any thoughts/suggestions?

We can do anything:) It would be easy to allow you to set the size in pixels, though I'm not sure this is what you want. Setting the size to 'OpenGL units' may be a bit harder. What does 1.0 in opengl space mean and how does that relate to point size? For one person it might mean scaling the font up, for someone else it may mean scaling down. Plus bitmaps and pixmaps have a pixel to pixel relationship that you can't change.

Here's some guidelines for vector and texture fonts. Take note that I say 'should' a lot :)

- One point in pixel space maps to 1 unit in OpenGL space, so a glyph that is 18 points high should be 18.0 units high.
- If you set an ortho projection to the window size and draw a glyph it's screen size should be the correct physical size ie a 72 point glyph on a 72dpi screen will be 1 inch high. Also if you set a perspective projection that maps 0.0 in the z axis to screen size you will get the same eg.

```
gluPerspective(90, window_height / 2 , small_number, large_number);
```

So basically it all depends on your projection matrix. Obviously you can use glScale but I understand if you don't want to.

Couple of extra things to note:

- The quality of vector glyphs will not change when you change the size, ie. a really small polygon glyph up close will look exactly the same as a big one from far away. They both contain the same amount of data. This doesn't apply to texture fonts.
- Secondly, there is a bug in the advance/kerning code that will cause ugliness at really small point sizes. This is because the advance and kerning use ints so an advance of 0.4 will become zero. If this is going to be a probelm, I can fix this.

Early on I did a lot of head scratching over the OpenGL unit to font size thing because when I was first integrating FTGL into my engine the fonts weren't the size I was expecting. I was tempted to build in some scaling but I decided doing nothing was the best approach because you can't please everyone. Plus it's 'correct' as it is.

Chapter 3

Projects using FTGL

To add your project to this list, please contact one of the FTGL developers at <http://sf.net/projects/ftgl>
Projects are listed in alphabetical order.

3.1 %FTGL language bindings

3.1.1 %FTGL#

FTGL# (<http://www.paskaluk.com/projects.php>) is a collection of .NET bindings for FTGL.

3.1.2 GIGuiA

GIGuiA (<http://sourceforge.net/projects/glguia/>) is a set of packages for Ada 2006 that can be used to create Graphical User Interfaces, relying (almost) only on OpenGL. Hence should be rather platform-independent.

3.1.3 Ruby %FTGL

Ruby FTGL# (<http://rubyforge.org/projects/ruby-ftgl/>) is a collection of Ruby bindings for FTGL.

3.1.4 PyFTGL

PyFTGL (<http://code.google.com/p/pyftgl/>) wraps the functionality of FTGL into a Python module so that it can be used in conjunction with PyOpenGL.

3.2 Projects currently using %FTGL

3.2.1 Agent World

Agent World (<http://code.google.com/p/agentw/>) provides tools for simulating and visualizing multi-agent systems and is specially designed for testing machine learning applications (and specially focused on Case Based Reasoning ones). It includes support for representing information using the Feature Term formalism, and provides a series of relational machine learning algorithms that can deal with them. The whole project is created in C++ to maximize efficiency, and uses OpenGL as the visualization library to ensure cross-platformness.

3.2.2 Amaltheia

Amaltheia (<http://home.gna.org/amaltheia/>) is a cross-platform game programming API that supports two backends, OpenGL and DirectX. The aim of the Amaltheia project is to create an intuitive and simple to use library, providing core 3d and 2d functionality in a platform independent manner. It also provides platform independence regarding basic network functions, input handling, threads and sound. Currently the GNU/Linux and the Windows OSes are supported.

3.2.3 Armagetron Advanced

Armagetron Advanced (<http://www.armagetronad.net/>) is a multiplayer game in 3d that attempts to emulate and expand on the lightcycle sequence from the movie Tron. It's an old school arcade game slung into the 21st century. Highlights include a customizable playing arena, HUD, unique graphics, and AI bots. For the more advanced player there are new game modes and a wide variety of physics settings to tweak as well.

3.2.4 Audicle

Audicle (<http://audicle.cs.princeton.edu/>) is an audio programming environment that integrates the programmability of the development environment with elements of the runtime environment. The result is a duct-taped intersection of a concurrent smart editor, compiler, virtual machine, and debugger.

3.2.5 Battlestar T.U.X.

Battlestar T.U.X. (<http://code.google.com/p/battlestar-tux/>) is a top-down scrolling shooter project.

3.2.6 BJS

BJS (<http://bjs.sourceforge.net/>) is a funny arcade 3D multiplayer tank battle. It is fully playable and very fun in multiplayer. Of course the single player is also possible. There is no story. You just get a tank and go shoot other players. Currently there are 5 different tanks, 6 maps, 9 powerups and 4 weapons.

3.2.7 Blender

Blender (<http://blender.org/>) is an integrated 3d suite for modelling, animation, rendering, post-production, interactive creation and playback (games).

3.2.8 Breve

Breve (<http://www.spiderland.org/>) is a free, open-source software package which makes it easy to build 3D simulations of multi-agent systems and artificial life. Using Python, or using a simple scripting language called *steve*, you can define the behaviors of agents in a 3D world and observe how they interact. *breve* includes physical simulation and collision detection so you can simulate realistic creatures, and an OpenGL display engine so you can visualize your simulated worlds.

3.2.9 BZFlag

BZFlag (<http://BZFlag.org/>) is a 3D multi-player multiplatform tank battle game that allows users to play against each other in a network environment.

BZFlag uses FTGL as of version 2.99.

3.2.10 Capture The Flag

Capture The Flag (<http://capturetf.sourceforge.net/>) is an open source, multi-platform, network game project.

3.2.11 Cello

Cello (<http://common-lisp.net/project/cello/>) is a project to create an open-source, industrial-strength, portable GUI toolkit for Common Lisp. Its features include anti-aliased fonts, accelerated 2d- and 3d-graphics, a standard set of GUI widgets, easy construction of new widgets, and much more. Cello heavily utilizes Cells (a sister project on common-lisp.net), in addition to industry-standard technologies such as OpenGL, FreeType, and ImageMagick.

3.2.12 Chimera

Chimera (<http://www.cgl.ucsf.edu/chimera/>) is a highly extensible program for interactive visualization and analysis of molecular structures and related data, including density maps, supramolecular assemblies, sequence alignments, docking results, trajectories, and conformational ensembles. High-quality images and animations can be generated.

3.2.13 Cinepaint

Cinepaint (<http://www.cinepaint.org/>) is a deep paint image retouching tool that supports higher color fidelity than ordinary painting tools.

3.2.14 Duel

Duel (<http://www.personal.rdg.ac.uk/~sir03me/play/code.html>) is a small overhead perspective spaceship game.

3.2.15 Empty Clip

Empty Clip (<http://emptyclip.sourceforge.net/>) is a top-down 2D Action RPG.

3.2.16 Freebox

Freebox (<http://freebox.sourceforge.net/>) is designed for use in a special type of computer called an 'HTPC', which is connected to a home-theatre system to watch XviD/DivX/DVD movies, play music (MP3, CD, whatever), play some emulated games, or whatever else you want to do with it.

3.2.17 Gem

Gem (<http://gem.iem.at/>) is a loadable library for puredata, which adds OpenGL graphics rendering and animation to Pd. Pd is a graphical programming language and computer music system.

3.2.18 GLMayan

GLMayan (<http://glmayan.sourceforge.net/>) is an OpenGL screensaver.

3.2.19 Glover

Glover (<http://code.google.com/p/glover/>) is a movie player that renders the content using OpenGL allowing all kinds of special effects using fragment shaders. The movie decoding is done using ffmpeg.

3.2.20 Ivf++

Ivf++ (<http://ivfplusplus.sourceforge.net/>) is a C++ library encapsulating OpenGL functionality. The primary goal is to make it easier to use the OpenGL library in interactive 3D applications. The second goal is extensibility, providing a set of well defined base classes for different object types to build new classes on. The third goal is portability, primarily between Linux and Windows, but the library should also be easily ported to Mac OS X.

3.2.21 Jahshaka

Jashaka (<http://jahshaka.org/>) is an advanced video editing, animation, visual effects, painting and music tool.

3.2.22 Karaoke FX

Karaoke FX (<http://jeanchristophe.duber.free.fr/karaokefx/>) is a midifile player that can display lyrics in synch with the sound so as it can be used for karaoke. It relies on plugins for midi output devices as for lyrics display.

3.2.23 Libinstrudeo

Libinstrudeo (<http://sourceforge.net/projects/libinstrudeo>), initially written for the ScreenKast program, provides the necessary logic to capture screen recordings and to process them. Includes a soap-client for the webservice at captorials.com that enables you to share your recordings.

3.2.24 Light Speed!

Light Speed! (<http://lightspeed.sourceforge.net/>) is an OpenGL-based program which illustrates the effects of special relativity on the appearance of moving objects. When an object accelerates past a few million meters per second, these effects begin to grow noticeable, becoming more and more pronounced as the speed of light is approached. These relativistic effects are viewpoint-dependent, and include shifts in length, object hue, brightness and shape.

3.2.25 MySQL GUI Tools

MySQL GUI Tools (<http://dev.mysql.com/downloads/gui-tools/5.0.html>) is a collection of tools for the MySQL database. It consists of MySQL Administrator, MySQL Query Browser and MySQL Migration Toolkit.

3.2.26 OctPlot

OctPlot (<http://octplot.sourceforge.net/>) is a graphics package for Octave, the free alternative to MATLAB. It provides high quality PostScript and on-screen graphics.

3.2.27 Open ActiveWrl

Open ActiveWrl (<http://open-activewrl.sourceforge.net/>) is a software development toolkit based on a generic software development approach that allows the implementation VRML/X3D browser components. These browser components can run within an conventional application or can be linked together for the implementation of parallel immersive VR setups.

3.2.28 OpenEagles

OpenEagles (<http://www.openeaagles.org/>) is a multi-platform simulation framework targeted to help simulation engineers and software developers build robust, scalable, virtual, constructive, stand-alone, and distributed simulation applications. It has been used extensively to build applications that demand real-time performance. This includes applications to conduct human factor studies, operator training, and the development of complete distributed virtual simulation systems. OpenEagles has also been used to build stand-alone and distributed constructive applications oriented at system analysis.

3.2.29 OpenGC

OpenGC (<http://www.opengc.org/>) is a multi-platform, multi-simulator, open-source C++ tool for developing and implementing high quality glass cockpit displays for simulated flightdecks.

3.2.30 OpenSG

OpenSG (<http://www.opensg.org/>) is a portable scenegraph system to create realtime graphics programs, e.g. for virtual reality applications.

3.2.31 Panthera

Panthera (<http://sourceforge.net/projects/panthera>) is a C++ framework for interactive visualization, manipulation, and editing of volume data. Applications developed on top of Panthera can utilize both desktop and immersive user interface devices, such as position trackers and haptic displays.

3.2.32 Planet Penguin Racer

PlanetPenguin Racer (<http://developer.berlios.de/projects/ppracer/>) is a simple OpenGL racing game featuring Tux, the Linux mascot. The goal of the game is to slide down a snow- and ice-covered mountain as quickly as possible, avoiding the trees and rocks that will slow you down.

3.2.33 projectM

projectM (<http://projectm.sourceforge.net/>) is a music visualizer which uses OpenGL for hardware acceleration. It is compatible with Milkdrop presets.

3.2.34 Puzzle Bobble 3D

Puzzle Bobble 3D (<http://homepage.mac.com/eric.lee/puzzle/>) is a 3D video game for Linux. The game is similar to Tetris/Connect 4: connect balls of the same colour to make them disappear. Puzzle Bobble 3D is based on an already popular arcade game of the same name by Taito Corporation (see links section at the bottom of this page), but this particular variant is played in a 3D environment (hence the name).

3.2.35 ROOT

ROOT (<http://root.cern.ch/>) is an object-oriented data analysis framework.

3.2.36 SCIRun

SCIRun (<http://software.sci.utah.edu/scirun.html>) is a Problem Solving Environment (PSE), for modeling, simulation and visualization of scientific problems. It is available for free and open source.

3.2.37 TINE

TINE, or TINE Is Not ELITE (<http://tine.sunsite.dk/en/index.html>) is an open source cross-platform remake of the classic space adventure game ELITE.

3.2.38 Tiny Planet

Tiny Planet (<http://www.duberga.net/tinyplanet/>) is a real-time OpenGL viewer of detailed earth texture such as BlueMarble from Earth Observatory (NASA) or any other planet texture. Vectorial data such as points of interest, boundaries, rivers can be superimposed to the texture.

3.2.39 Truevision

Truevision (<http://truevision.sourceforge.net/>) is a 3D modeler for GNOME.

3.2.40 Tulip

Tulip (<http://tulip.labri.fr/>) is a system dedicated to the visualization of huge graphs. It is capable of managing graphs with up to 500,000 nodes and edges on relatively modest hardware (eg. 600MHz Pentium III, 256MB RAM).

3.2.41 Ubit

Ubit (<http://www.infres.enst.fr/~elc/ubit/>) Ubit is a new GUI toolkit that combines the advantages of scene graph and widget based toolkits. The Ubit3D extension makes it possible to display 2D GUIs in a 3D space.

3.2.42 VRS

The Virtual Rendering System (<http://www.hpi.uni-potsdam.de/vrs/>) is a computer graphics software library for constructing interactive 3D applications. It provides a large collection of 3D rendering components which facilitate implementing 3D graphics applications and experimenting with 3D graphics and imaging algorithms.

3.2.43 VTK

VTK, the Visualization Toolkit (<http://www.vtk.org/>), is an object oriented, high level library that allows one to easily write C++ programs, Tcl, Python and Java scripts that do 3D visualization.

3.2.44 XLock

XLock (<http://www.tux.org/~bagleyd/xlockmore.html>) is a screensaver and screen locking utility with additional OpenGL and XPM modes.

3.3 Projects that used to use %FTGL

3.3.1 GNU Backgammon

GNU Backgammon (<http://www.gnubg.org/>) was using FTGL until version 0.14.3+20060520-1.

3.3.2 OpenSceneGraph

OpenSceneGraph (<http://www.openscenegraph.org/projects/osg>) is an open source high performance 3D graphics toolkit, used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modelling. Written entirely in Standard C++ and OpenGL it runs on all Windows platforms, OSX, GNU/Linux, IRIX, Solaris, HP-Ux, AIX and FreeBSD operating systems.

3.3.3 Teddy

Teddy (<http://teddy.sourceforge.net/>) was a 3D graphics library. The main purpose was to be a simple scene graph manager.

3.3.4 VigiPac

VigiPac (<http://vigipac.sourceforge.net/>) was a three-dimensional Pacman clone with multiplayer support, written in the C++ language.

Chapter 4

FTGL tutorial

4.1 Starting to use %FTGL

Only one header is required to use FTGL:

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

4.2 Choosing a font type

FTGL supports 6 font output types among 3 groups: raster fonts, vector fonts, and texture fonts which are a mixture of both. Each font type has its advantages and disadvantages.

4.2.1 Raster fonts

Raster fonts are made of pixels painted directly on the viewport's framebuffer. They cannot be directly rotated or scaled.

- Bitmap fonts use 1-bit (2-colour) rasterised glyphs.
- Pixmap fonts use 8-bit (256 levels) rasterised glyphs.

This is a GLBitmapFont object.
This is a GLPixmapFont object.

4.2.2 Vector fonts

Vector fonts are 3D objects that are rendered at the current matrix location. All position, scale, texture and material effects apply to vector fonts.

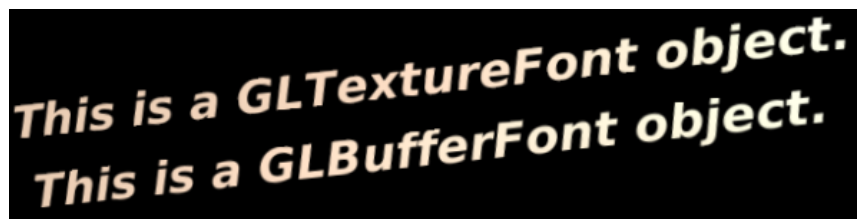
- Polygon fonts use planar triangle meshes and can be texture-mapped.
- Outline fonts use OpenGL lines.
- Extruded fonts are extruded polygon fonts, with the front, back and side meshes renderable separately to apply different effects and materials.



4.2.3 Textured fonts

Textured fonts are probably the most versatile types. They are fast, antialiased, and can be transformed just like any OpenGL primitive.

- Texture fonts use one texture per glyph. They are fast because glyphs are stored permanently in the video card's memory.
- Buffer fonts use one texture per line of text. They tend to be faster than texture fonts when the same line of text needs to be rendered for more than one frame.



4.3 Create font objects

Creating a font and displaying some text is really straightforward, be it in C or in C++.

4.3.1 in C

```

/* Create a pixmap font from a TrueType file. */
FTGLFont *font = ftglCreatePixmapFont("/home/user/Arial.ttf");

/* If something went wrong, bail out. */
if(!font)
    return -1;

/* Set the font size and render a small text. */
ftglSetFontFaceSize(font, 72, 72);
ftglRenderFont(font, "Hello World!", FTGL_RENDER_ALL);

/* Destroy the font object. */
ftglDestroyFont(font);

```

4.3.2 in C++

```

// Create a pixmap font from a TrueType file.
FTGLPixmapFont font("/home/user/Arial.ttf");

// If something went wrong, bail out.
if(font.Error())
    return -1;

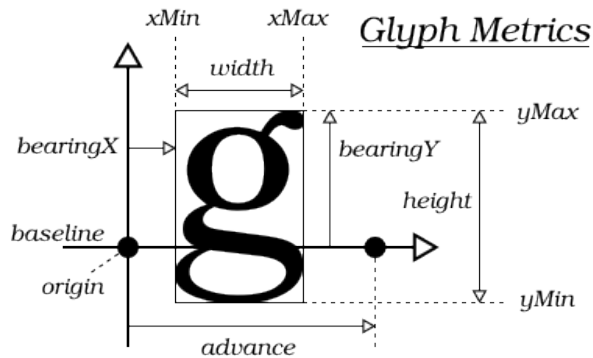
// Set the font size and render a small text.
font.FaceSize(72);
font.Render("Hello World!");

```

The first 128 glyphs of the font (generally corresponding to the ASCII set) are preloaded. This means that usual text is rendered fast enough, but no memory is wasted loading glyphs that will not be used.

4.4 More font commands

4.4.1 Font metrics



If you ask a font to render at 0.0, 0.0 the bottom left most pixel or polygon may not be aligned to 0.0, 0.0. With `Font::Ascender()` (p. 39), `Font::Descender()` (p. 42) and `Font::Advance()` (p. 38) an approximate bounding box can be calculated.

For an exact bounding box, use the `Font::BBox()` (p. 40) function. This function returns the extent of the volume containing 'string'. 0.0 on the y axis will be aligned with the font baseline.

4.4.2 Specifying a character map encoding

From the FreeType documentation:

"By default, when a new face object is created, (FreeType) lists all the charmaps contained in the font face and selects the one that supports Unicode character codes if it finds one. Otherwise, it tries to find support for Latin-1, then ASCII."

It then gives up. In this case FTGL will set the charmap to the first it finds in the fonts charmap list. You can explicitly set the char encoding with `Font::CharMap()` (p. 41).

Valid encodings as of FreeType 2.0.4 are:

- `ft_encoding_none`
- `ft_encoding_unicode`
- `ft_encoding_symbol`
- `ft_encoding_latin_1`
- `ft_encoding_latin_2`
- `ft_encoding_sjis`
- `ft_encoding_gb2312`
- `ft_encoding_big5`
- `ft_encoding_wansung`
- `ft_encoding_johab`
- `ft_encoding_adobe_standard`

- `ft_encoding_adobe_expert`
- `ft_encoding_adobe_custom`
- `ft_encoding_apple_roman`

For instance:

```
font.CharMap(ft_encoding_apple_roman);
```

This will return an error if the requested encoding can't be found in the font.

If your application uses Latin-1 characters, you can preload this character set using the following code:

```
// Create a pixmap font from a TrueType file.
FTGLPixmapFont font("/home/user/Arial.ttf");

// If something went wrong, bail out.
if(font.Error())
    return -1;

// Set the face size and the character map. If something went wrong, bail out.
font.FaceSize(72);
if(!font.CharMap(ft_encoding_latin_1))
    return -1;

// Create a string containing all characters between 128 and 255
// and preload the Latin-1 chars without rendering them.
char buf[129];
for(int i = 128; i < 256; i++)
{
    buf[i] = (char)(unsigned char)i;
}
buf[128] = '\0';

font.Advance(buf);
}
```

4.5 Sample font manager class

```
FTTextureFont* myFont = FTGLFontManager::Instance().GetFont("arial.ttf", 72);

#include <map>
#include <string>
#include <FTGL/ftgl.h>

using namespace std;

typedef map<string, FTFont*> FontList;
typedef FontList::const_iterator FontIter;

class FTGLFontManager
{
public:
    // NOTE
    // This is shown here for brevity. The implementation should be in the source
    // file otherwise your compiler may inline the function resulting in
    // multiple instances of FTGLFontManager
    static FTGLFontManager& Instance()
    {
        static FTGLFontManager tm;
        return tm;
    }

    ~FTGLFontManager()
    {
        FontIter font;
        for(font = fonts.begin(); font != fonts.end(); font++)
        {
            delete (*font).second;
        }

        fonts.clear();
    }

    FTFont* GetFont(const char *filename, int size)
```

```
{
    char buf[256];
    sprintf(buf, "%s%i", filename, size);
    string fontKey = string(buf);

    FontIter result = fonts.find(fontKey);
    if(result != fonts.end())
    {
        LOGMSG("Found font %s in list", filename);
        return result->second;
    }

    FTFont* font = new FTTextureFont;

    string fullname = path + string(filename);

    if(!font->Open(fullname.c_str()))
    {
        LOGERROR("Font %s failed to open", fullname.c_str());
        delete font;
        return NULL;
    }

    if(!font->FaceSize(size))
    {
        LOGERROR("Font %s failed to set size %i", filename, size);
        delete font;
        return NULL;
    }

    fonts[fontKey] = font;

    return font;
}

private:
    // Hide these 'cause this is a singleton.
    FTGLFontManager(){}
    FTGLFontManager(const FTGLFontManager&){};
    FTGLFontManager& operator = (const FTGLFontManager&){ return *this; };

    // container for fonts
    FontList fonts;
};
```


Chapter 5

Namespace Documentation

5.1 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 }

5.1.1 Enumeration Type Documentation

5.1.1.1 enum FTGL::RenderMode

Enumerator

RENDER_FRONT
RENDER_BACK
RENDER_SIDE
RENDER_ALL

Definition at line 53 of file ftgl.h.

5.1.1.2 enum FTGL::TextAlignment

Enumerator

ALIGN_LEFT
ALIGN_CENTER
ALIGN_RIGHT
ALIGN_JUSTIFY

Definition at line 61 of file ftgl.h.

Chapter 6

Data Structure Documentation

6.1 FTBBox Class Reference

FTBBox (p. 21) is a convenience class for handling bounding boxes.

```
#include <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

6.1.1 Detailed Description

FTBBox (p. 21) is a convenience class for handling bounding boxes.

Definition at line 42 of file FTBBox.h.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 FTBBBox::FTBBBox () [inline]

Default constructor.

Bounding box is set to zero.

Definition at line 48 of file FTBBBox.h.

6.1.2.2 FTBBBox::FTBBBox (float *lx*, float *ly*, float *lz*, float *ux*, float *uy*, float *uz*) [inline]

Constructor.

Definition at line 56 of file FTBBBox.h.

6.1.2.3 FTBBBox::FTBBBox (FTPoint *l*, FTPoint *u*) [inline]

Constructor.

Definition at line 64 of file FTBBBox.h.

6.1.2.4 FTBBBox::FTBBBox (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

<i>glyph</i>	A freetype glyph
--------------	------------------

Definition at line 75 of file FTBBBox.h.

6.1.2.5 FTBBBox::~FTBBBox () [inline]

Destructor.

Definition at line 93 of file FTBBBox.h.

6.1.3 Member Function Documentation

6.1.3.1 void FTBBBox::Invalidate () [inline]

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

Definition at line 100 of file FTBBBox.h.

6.1.3.2 bool FTBBBox::IsValid () [inline]

Determines if this bounding box is valid.

Returns

True if all lower values are \leq the corresponding upper values.

Definition at line 112 of file FTBBBox.h.

6.1.3.3 FTPoint const FTBox::Lower () const [inline]

Definition at line 165 of file FTBox.h.

Referenced by FTFont::BBox().

6.1.3.4 FTBox& FTBox::operator+=(const FTPoint vector) [inline]

Move the Bounding Box by a vector.

Parameters

<i>vector</i>	The vector to move the bbox in 3D space.
---------------	--

Definition at line 124 of file FTBox.h.

6.1.3.5 FTBox& FTBox::operator|= (const FTBox & bbox) [inline]

Combine two bounding boxes.

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

Parameters

<i>bbox</i>	The bounding box to merge with the second one.
-------------	--

Definition at line 138 of file FTBox.h.

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

6.1.3.6 void FTBox::SetDepth (float depth) [inline]

Definition at line 150 of file FTBox.h.

6.1.3.7 FTPoint const FTBox::Upper () const [inline]

Definition at line 159 of file FTBox.h.

Referenced by FTFont::BBox().

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

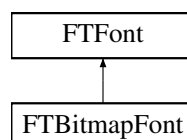
- **FTBox.h**

6.2 FTBitmapFont Class Reference

FTBitmapFont (p. 23) is a specialisation of the **FTFont** (p. 36) class for handling Bitmap fonts.

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

Inheritance diagram for FTBitmapFont:



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.

6.2.1 Detailed Description

FTBitmapFont (p. 23) is a specialisation of the **FTFont** (p. 36) class for handling Bitmap fonts.

See also

FTFont (p. 36)

Definition at line 45 of file FTGLBitmapFont.h.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 FTBitmapFont::FTBitmapFont (const char * fontFilePath)

Open and read a font file.

Sets Error flag.

Parameters

<i>fontFilePath</i>	font file path.
---------------------	-----------------

6.2.2.2 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. 19). The pointer must be valid while using **FTGL** (p. 19).

Parameters

<i>pBufferBytes</i>	the in-memory buffer
<i>bufferSizeInBytes</i>	the length of the buffer in bytes

6.2.2.3 FTBitmapFont::~FTBitmapFont ()

Destructor.

6.2.3 Member Function Documentation

6.2.3.1 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. 45).

Parameters

<i>slot</i>	A FreeType glyph slot.
-------------	------------------------

Returns

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

Implements **FTFont** (p. 43).

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

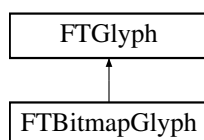
- **FTGLBitmapFont.h**

6.3 FTBitmapGlyph Class Reference

FTBitmapGlyph (p. 25) is a specialisation of **FTGlyph** (p. 45) for creating bitmaps.

```
#include <FTBitmapGlyph.h>
```

Inheritance diagram for FTBitmapGlyph:



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.

Additional Inherited Members

6.3.1 Detailed Description

FTBitmapGlyph (p. 25) is a specialisation of **FTGlyph** (p. 45) for creating bitmaps.

Definition at line 42 of file FTBitmapGlyph.h.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 FTBitmapGlyph::FTBitmapGlyph (FT_GlyphSlot *glyph*)

Constructor.

Parameters

<i>glyph</i>	The Freetype glyph to be processed
--------------	------------------------------------

6.3.2.2 virtual FTBitmapGlyph::~FTBitmapGlyph () [virtual]

Destructor.

6.3.3 Member Function Documentation

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

Render this glyph at the current pen position.

Parameters

<i>pen</i>	The current pen position.
<i>renderMode</i>	Render mode to display

Returns

The advance distance for this glyph.

Implements **FTGlyph** (p. 47).

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

- **FTBitmapGlyph.h**

6.4 FTBuffer Class Reference

FTBuffer (p. 27) is a helper class for pixel buffers.

```
#include <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.

6.4.1 Detailed Description

FTBuffer (p. 27) is a helper class for pixel buffers.

It provides the interface between **FTBufferFont** (p. 30) and **FTBufferGlyph** (p. 32) to optimise rendering operations.

See also

FTBufferGlyph (p. 32)

FTBufferFont (p. 30)

Definition at line 45 of file FTBuffer.h.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 FTBuffer::FTBuffer ()

Default constructor.

6.4.2.2 FTBuffer::~~FTBuffer ()

Destructor.

6.4.3 Member Function Documentation

6.4.3.1 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.

6.4.3.2 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.

6.4.3.3 FTPoint FTBuffer::Pos () const [inline]

Get the pen's position in the buffer.

Returns

The pen's position as an **FTPoint** (p. 56) object.

Definition at line 63 of file FTBuffer.h.

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

Set the pen's position in the buffer.

Parameters

<i>arg</i>	An FTPoint (p. 56) object with the desired pen's position.
------------	---

Definition at line 73 of file FTBuffer.h.

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

Set the buffer's size.

Parameters

<i>w</i>	The buffer's desired width, in pixels.
<i>h</i>	The buffer's desired height, in pixels.

6.4.3.6 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.

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

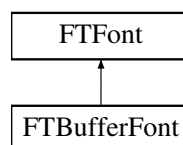
- **FTBuffer.h**

6.5 FTBufferFont Class Reference

FTBufferFont (p. 30) is a specialisation of the **FTFont** (p. 36) class for handling memory buffer fonts.

```
#include <FTBufferFont.h>
```

Inheritance diagram for FTBufferFont:



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.

6.5.1 Detailed Description

FTBufferFont (p. 30) is a specialisation of the **FTFont** (p. 36) class for handling memory buffer fonts.

See also

FTFont (p. 36)

Definition at line 43 of file FTBufferFont.h.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 FTBufferFont::FTBufferFont (const char * *fontFilePath*)

Open and read a font file.

Sets Error flag.

Parameters

<i>fontFilePath</i>	font file path.
---------------------	-----------------

6.5.2.2 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. 19). The pointer must be valid while using **FTGL** (p. 19).

Parameters

<i>pBufferBytes</i>	the in-memory buffer
<i>bufferSizeInBytes</i>	the length of the buffer in bytes

6.5.2.3 FTBufferFont::~~FTBufferFont ()

Destructor.

6.5.3 Member Function Documentation

6.5.3.1 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. 45).

Parameters

<i>slot</i>	A FreeType glyph slot.
-------------	------------------------

Returns

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

Implements **FTFont** (p. 43).

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

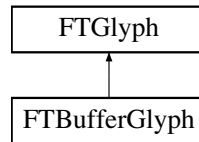
- **FTBufferFont.h**

6.6 FTBufferGlyph Class Reference

FTBufferGlyph (p. 32) is a specialisation of **FTGlyph** (p. 45) for memory buffer rendering.

```
#include <FTBufferGlyph.h>
```

Inheritance diagram for FTBufferGlyph:



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.

Additional Inherited Members

6.6.1 Detailed Description

FTBufferGlyph (p. 32) is a specialisation of **FTGlyph** (p. 45) for memory buffer rendering.

Definition at line 40 of file FTBufferGlyph.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 FTBufferGlyph::FTBufferGlyph (FT_GlyphSlot glyph, FTBuffer * buffer)

Constructor.

Parameters

<i>glyph</i>	The Freetype glyph to be processed
<i>buffer</i>	An FTBuffer (p. 27) object in which to render the glyph.

6.6.2.2 virtual FTBufferGlyph::~FTBufferGlyph () [virtual]

Destructor.

6.6.3 Member Function Documentation

6.6.3.1 virtual const FTPoint& FTBufferGlyph::Render (const FTPoint & pen, int renderMode) [virtual]

Render this glyph at the current pen position.

Parameters

<i>pen</i>	The current pen position.
<i>renderMode</i>	Render mode to display

Returns

The advance distance for this glyph.

Implements **FTGlyph** (p. 47).

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

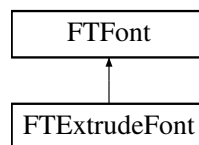
- **FTBufferGlyph.h**

6.7 FTExtrudeFont Class Reference

FTExtrudeFont (p. 33) is a specialisation of the **FTFont** (p. 36) class for handling extruded Polygon fonts.

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

Inheritance diagram for FTExtrudeFont:



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.

6.7.1 Detailed Description

FTExtrudeFont (p. 33) is a specialisation of the **FTFont** (p. 36) class for handling extruded Polygon fonts.

See also

- FTFont** (p. 36)
- FTPolygonFont** (p. 62)

Definition at line 46 of file FTGLExtrdFont.h.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 FTExtrudeFont::FTExtrudeFont (const char * *fontFilePath*)

Open and read a font file.

Sets Error flag.

Parameters

<i>fontFilePath</i>	font file path.
---------------------	-----------------

6.7.2.2 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. 19). The pointer must be valid while using **FTGL** (p. 19).

Parameters

<i>pBufferBytes</i>	the in-memory buffer
<i>bufferSizeInBytes</i>	the length of the buffer in bytes

6.7.2.3 FTExtrudeFont::~~FTExtrudeFont ()

Destructor.

6.7.3 Member Function Documentation

6.7.3.1 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. 45).

Parameters

<i>slot</i>	A FreeType glyph slot.
-------------	------------------------

Returns

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

Implements **FTFont** (p. 43).

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

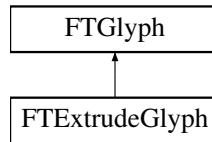
- **FTGLExtrdFont.h**

6.8 FTExtrudeGlyph Class Reference

FTExtrudeGlyph (p. 34) is a specialisation of **FTGlyph** (p. 45) for creating tessellated extruded polygon glyphs.

```
#include <FTExtrdGlyph.h>
```

Inheritance diagram for FTExtrudeGlyph:



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.

Additional Inherited Members

6.8.1 Detailed Description

FTExtrudeGlyph (p. 34) is a specialisation of **FTGlyph** (p. 45) for creating tessellated extruded polygon glyphs. Definition at line 43 of file FTExtrdGlyph.h.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 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

<i>glyph</i>	The Freetype glyph to be processed
<i>depth</i>	The distance along the z axis to extrude the glyph
<i>frontOutset</i>	outset contour size
<i>backOutset</i>	outset contour size
<i>useDisplayList</i>	Enable or disable the use of Display Lists for this glyph <code>true</code> turns ON display lists. <code>false</code> turns OFF display lists.

6.8.2.2 virtual FTExtrudeGlyph::~~FTExtrudeGlyph () [virtual]

Destructor.

6.8.3 Member Function Documentation

6.8.3.1 virtual const FTPoint& FTExtrudeGlyph::Render (const FTPoint & pen, int renderMode) [virtual]

Render this glyph at the current pen position.

Parameters

<i>pen</i>	The current pen position.
<i>renderMode</i>	Render mode to display

Returns

The advance distance for this glyph.

Implements **FTGlyph** (p. 47).

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

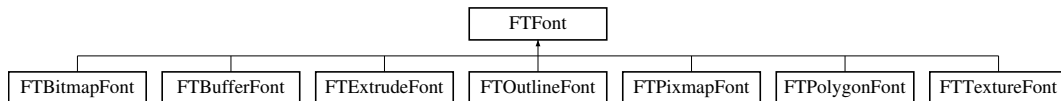
- **FTEXtrdGlyph.h**

6.9 FTFont Class Reference

FTFont (p. 36) is the public interface for the **FTGL** (p. 19) library.

```
#include <FTFont.h>
```

Inheritance diagram for FTFont:



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. 19).
- 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**(), **FTPoint** 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**(), **FTPoint** 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**(), **FTPoint** 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**

6.9.1 Detailed Description

FTFont (p. 36) is the public interface for the **FTGL** (p. 19) 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. 42). Check the freetype file `fterrdef.h` for error definitions.

See also

`FTFace`
`FTSize`

Definition at line 56 of file `FTFont.h`.

6.9.2 Constructor & Destructor Documentation

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

Open and read a font file.

Sets Error flag.

Parameters

<code>fontFilePath</code>	font file path.
---------------------------	-----------------

6.9.2.2 `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. 19). The pointer must be valid while using **FTGL** (p. 19).

Parameters

<code>pBufferBytes</code>	the in-memory buffer
<code>bufferSizeInBytes</code>	the length of the buffer in bytes

6.9.2.3 `virtual FTFont::~FTFont ()` [virtual]

6.9.3 Member Function Documentation

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

Get the advance for a string.

Parameters

<code>string</code>	'C' style string to be checked.
---------------------	---------------------------------

<i>len</i>	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
<i>spacing</i>	A displacement vector to add after each character has been checked (optional).

Returns

The string's advance width.

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

Get the advance for a string.

Parameters

<i>string</i>	A wchar_t string
<i>len</i>	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
<i>spacing</i>	A displacement vector to add after each character has been checked (optional).

Returns

The string's advance width.

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

Get the global ascender height for the face.

Returns

Ascender height

6.9.3.4 `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

<i>fontFilePath</i>	auxilliary font file path.
---------------------	----------------------------

Returns

`true` if file has been attached successfully.

6.9.3.5 `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

<i>pBufferBytes</i>	the in-memory buffer.
<i>bufferSizeInBytes</i>	the length of the buffer in bytes.

Returns

`true` if file has been attached successfully.

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

Get the bounding box for a string.

Parameters

<i>string</i>	A char buffer.
<i>len</i>	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
<i>position</i>	The pen position of the first character (optional).
<i>spacing</i>	A displacement vector to add after each character has been checked (optional).

Returns

The corresponding bounding box.

Referenced by `BBox()`.

6.9.3.7 `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

<i>string</i>	A char buffer.
<i>llx</i>	Lower left near x coordinate.
<i>lly</i>	Lower left near y coordinate.
<i>llz</i>	Lower left near z coordinate.
<i>urx</i>	Upper right far x coordinate.
<i>ury</i>	Upper right far y coordinate.
<i>urz</i>	Upper right far z coordinate.

Definition at line 251 of file `FTFont.h`.

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

6.9.3.8 `virtual FTBBBox 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

<i>string</i>	A <code>wchar_t</code> buffer.
<i>len</i>	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
<i>position</i>	The pen position of the first character (optional).
<i>spacing</i>	A displacement vector to add after each character has been checked (optional).

Returns

The corresponding bounding box.

6.9.3.9 `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

<i>string</i>	A <code>wchar_t</code> buffer.
<i>llx</i>	Lower left near x coordinate.
<i>lly</i>	Lower left near y coordinate.
<i>llz</i>	Lower left near z coordinate.
<i>urx</i>	Upper right far x coordinate.
<i>ury</i>	Upper right far y coordinate.
<i>urz</i>	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()`.

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

Set the character map for the face.

Parameters

<i>encoding</i>	Freetype enumerate for char map code.
-----------------	---------------------------------------

Returns

`true` if charmap was valid and set correctly.

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

Get the number of character maps in this face.

Returns

character map count.

6.9.3.12 `virtual FT_Encoding* FTFont::CharMapList ()` `[virtual]`

Get a list of character maps in this face.

Returns

pointer to the first encoding.

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

Set the extrusion distance for the font.

Only implemented by **FTExtrudeFont** (p. 33)

Parameters

<i>depth</i>	The extrusion distance.
--------------	-------------------------

6.9.3.14 virtual float FTFont::Descender () const [virtual]

Gets the global descender height for the face.

Returns

Descender height

6.9.3.15 virtual FT_Error FTFont::Error () const [virtual]

Queries the Font for errors.

Returns

The current error code.

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

Set the char size for the current face.

Parameters

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

Returns

`true` if size was set correctly

6.9.3.17 virtual unsigned int FTFont::FaceSize () const [virtual]

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

Returns

face size

6.9.3.18 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

<i>flags</i>	The glyph loading flags.
--------------	--------------------------

6.9.3.19 virtual float FTFont::LineHeight () const [virtual]

Gets the line spacing for the font.

Returns

Line height

6.9.3.20 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. 45).

Parameters

<i>slot</i>	A FreeType glyph slot.
-------------	------------------------

Returns

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

Implemented in **FTEXtrudeFont** (p. 34), **FTBitmapFont** (p. 25), **FTOutlineFont** (p. 52), **FTPixmapFont** (p. 55), **FTPolygonFont** (p. 63), **FTTextureFont** (p. 71), and **FTBufferFont** (p. 31).

6.9.3.21 virtual void FTFont::Outset (float outset) [virtual]

Set the outset distance for the font.

Only implemented by **FTOutlineFont** (p. 51), **FTPolygonFont** (p. 62) and **FTEXtrudeFont** (p. 33)

Parameters

<i>outset</i>	The outset distance.
---------------	----------------------

6.9.3.22 virtual void FTFont::Outset (float front, float back) [virtual]

Set the front and back outset distances for the font.

Only implemented by **FTEXtrudeFont** (p. 33)

Parameters

<i>front</i>	The front outset distance.
<i>back</i>	The back outset distance.

6.9.3.23 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

<i>string</i>	'C' style string to be output.
<i>len</i>	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
<i>position</i>	The pen position of the first character (optional).
<i>spacing</i>	A displacement vector to add after each character has been displayed (optional).
<i>renderMode</i>	Render mode to use for display (optional).

Returns

The new pen position after the last character was output.

```
6.9.3.24 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

<i>string</i>	wchar_t string to be output.
<i>len</i>	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
<i>position</i>	The pen position of the first character (optional).
<i>spacing</i>	A displacement vector to add after each character has been displayed (optional).
<i>renderMode</i>	Render mode to use for display (optional).

Returns

The new pen position after the last character was output.

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

Enable or disable the use of Display Lists inside **FTGL** (p. 19).

Parameters

<i>useList</i>	true turns ON display lists. false turns OFF display lists.
----------------	---

6.9.4 Friends And Related Function Documentation

```
6.9.4.1 friend class FTBitmapFont [friend]
```

Definition at line 78 of file FTFont.h.

```
6.9.4.2 friend class FTBufferFont [friend]
```

Definition at line 79 of file FTFont.h.

```
6.9.4.3 friend class FTExtrudeFont [friend]
```

Definition at line 80 of file FTFont.h.

6.9.4.4 friend class FTFontImpl [friend]

Definition at line 367 of file FTFont.h.

6.9.4.5 friend class FTOutlineFont [friend]

Definition at line 81 of file FTFont.h.

6.9.4.6 friend class FTPixmapFont [friend]

Definition at line 82 of file FTFont.h.

6.9.4.7 friend class FTPolygonFont [friend]

Definition at line 83 of file FTFont.h.

6.9.4.8 friend class FTTextureFont [friend]

Definition at line 84 of file FTFont.h.

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

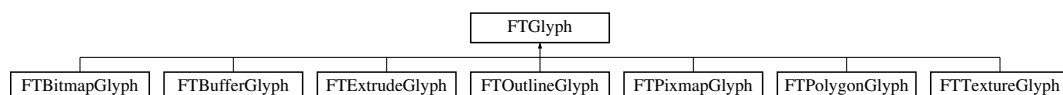
- **FTFont.h**

6.10 FTGlyph Class Reference

FTGlyph (p. 45) is the base class for **FTGL** (p. 19) glyphs.

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

Inheritance diagram for FTGlyph:



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**

6.10.1 Detailed Description

FTGlyph (p. 45) is the base class for **FTGL** (p. 19) 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. 21)

FTPoint (p. 56)

Definition at line 50 of file FTGlyph.h.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 FTGlyph::FTGlyph (FT_GlyphSlot glyph) [protected]

Create a glyph.

Parameters

<i>glyph</i>	The FreeType glyph to be processed
--------------	------------------------------------

6.10.2.2 virtual FTGlyph::~~FTGlyph () [virtual]

Destructor.

6.10.3 Member Function Documentation

6.10.3.1 virtual float FTGlyph::Advance () const [virtual]

Return the advance width for this glyph.

Returns

advance width.

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

Return the bounding box for this glyph.

Returns

bounding box.

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

Queries for errors.

Returns

The current error code.

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

Renders this glyph at the current pen position.

Parameters

<i>pen</i>	The current pen position.
<i>renderMode</i>	Render mode to display

Returns

The advance distance for this glyph.

Implemented in **FTExtrudeGlyph** (p. 35), **FTTextureGlyph** (p. 72), **FTPolygonGlyph** (p. 65), **FTOutlineGlyph** (p. 53), **FTBitmapGlyph** (p. 27), **FTPixmapGlyph** (p. 56), and **FTBufferGlyph** (p. 32).

6.10.4 Friends And Related Function Documentation

6.10.4.1 `friend class FTBitmapGlyph` [friend]

Definition at line 70 of file FTGlyph.h.

6.10.4.2 `friend class FTBufferGlyph` [friend]

Definition at line 71 of file FTGlyph.h.

6.10.4.3 `friend class FTExtrudeGlyph` [friend]

Definition at line 72 of file FTGlyph.h.

6.10.4.4 `friend class FTOutlineGlyph` [friend]

Definition at line 73 of file FTGlyph.h.

6.10.4.5 `friend class FTPixmapGlyph` [friend]

Definition at line 74 of file FTGlyph.h.

6.10.4.6 friend class **FTPolygonGlyph** [friend]

Definition at line 75 of file FTGlyph.h.

6.10.4.7 friend class **FTTextureGlyph** [friend]

Definition at line 76 of file FTGlyph.h.

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

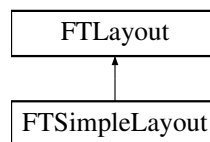
- **FTGlyph.h**

6.11 FTLayout Class Reference

FTLayout (p. 48) is the interface for layout managers that render text.

```
#include <FTLayout.h>
```

Inheritance diagram for FTLayout:



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**

6.11.1 Detailed Description

FTLayout (p. 48) 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. 36)

FTBBox (p. 21)

Definition at line 52 of file FTLayout.h.

6.11.2 Constructor & Destructor Documentation

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

6.11.2.2 `virtual FTLayout::~~FTLayout ()` [virtual]

Destructor.

6.11.3 Member Function Documentation

6.11.3.1 `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

<i>string</i>	A char string.
<i>len</i>	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
<i>position</i>	The pen position of the first character (optional).

Returns

The corresponding bounding box.

Implemented in **FTSimpleLayout** (p. 67).

6.11.3.2 `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

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

<i>position</i>	The pen position of the first character (optional).
-----------------	---

Returns

The corresponding bounding box.

Implemented in **FTSimpleLayout** (p. 67).

6.11.3.3 virtual FT_Error FTLayou::Error () const [virtual]

Queries the Layout for errors.

Returns

The current error code.

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

Render a string of characters.

Parameters

<i>string</i>	'C' style string to be output.
<i>len</i>	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
<i>position</i>	The pen position of the first character (optional).
<i>renderMode</i>	Render mode to display (optional)

Implemented in **FTSimpleLayout** (p. 68).

6.11.3.5 virtual void FTLayou::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

<i>string</i>	wchar_t string to be output.
<i>len</i>	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
<i>position</i>	The pen position of the first character (optional).
<i>renderMode</i>	Render mode to display (optional)

Implemented in **FTSimpleLayout** (p. 68).

6.11.4 Friends And Related Function Documentation

6.11.4.1 friend class FTSimpleLayout [friend]

Definition at line 67 of file FTLayou.h.

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

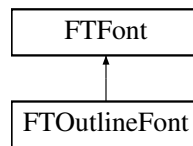
- **FTLayout.h**

6.12 FTOutlineFont Class Reference

FTOutlineFont (p. 51) is a specialisation of the **FTFont** (p. 36) class for handling Vector Outline fonts.

```
#include <FTGLOutlineFont.h>
```

Inheritance diagram for FTOutlineFont:



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.

6.12.1 Detailed Description

FTOutlineFont (p. 51) is a specialisation of the **FTFont** (p. 36) class for handling Vector Outline fonts.

See also

FTFont (p. 36)

Definition at line 45 of file FTGLOutlineFont.h.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 FTOutlineFont::FTOutlineFont (const char * fontFilePath)

Open and read a font file.

Sets Error flag.

Parameters

<i>fontFilePath</i>	font file path.
---------------------	-----------------

6.12.2.2 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. 19). The pointer must be valid while using **FTGL** (p. 19).

Parameters

<i>pBufferBytes</i>	the in-memory buffer
<i>bufferSizeInBytes</i>	the length of the buffer in bytes

6.12.2.3 FTOutlineFont::~FTOutlineFont ()

Destructor.

6.12.3 Member Function Documentation

6.12.3.1 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. 45).

Parameters

<i>slot</i>	A FreeType glyph slot.
-------------	------------------------

Returns

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

Implements **FTFont** (p. 43).

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

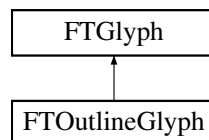
- **FTGLOutlineFont.h**

6.13 FTOutlineGlyph Class Reference

FTOutlineGlyph (p. 52) is a specialisation of **FTGlyph** (p. 45) for creating outlines.

```
#include <FTOutlineGlyph.h>
```

Inheritance diagram for FTOutlineGlyph:



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.

Additional Inherited Members

6.13.1 Detailed Description

FTOutlineGlyph (p. 52) is a specialisation of **FTGlyph** (p. 45) for creating outlines.

Definition at line 42 of file FTOutlineGlyph.h.

6.13.2 Constructor & Destructor Documentation

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

Constructor.

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

Parameters

<i>glyph</i>	The Freetype glyph to be processed
<i>outset</i>	outset distance
<i>useDisplayList</i>	Enable or disable the use of Display Lists for this glyph <code>true</code> turns ON display lists. <code>false</code> turns OFF display lists.

6.13.2.2 virtual FTOutlineGlyph::~FTOutlineGlyph () [virtual]

Destructor.

6.13.3 Member Function Documentation

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

Render this glyph at the current pen position.

Parameters

<i>pen</i>	The current pen position.
<i>renderMode</i>	Render mode to display

Returns

The advance distance for this glyph.

Implements **FTGlyph** (p. 47).

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

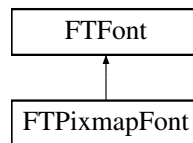
- **FTOutlineGlyph.h**

6.14 FTPixmapFont Class Reference

FTPixmapFont (p. 53) is a specialisation of the **FTFont** (p. 36) class for handling Pixmap (Grey Scale) fonts.

```
#include <FTGLPixmapFont.h>
```

Inheritance diagram for FTPixmapFont:



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.

6.14.1 Detailed Description

FTPixmapFont (p. 53) is a specialisation of the **FTFont** (p. 36) class for handling Pixmap (Grey Scale) fonts.

See also

FTFont (p. 36)

Definition at line 45 of file FTGLPixmapFont.h.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 FTPixmapFont::FTPixmapFont (const char * fontFilePath)

Open and read a font file.

Sets Error flag.

Parameters

<i>fontFilePath</i>	font file path.
---------------------	-----------------

6.14.2.2 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. 19). The pointer must be valid while using **FTGL** (p. 19).

Parameters

<i>pBufferBytes</i>	the in-memory buffer
<i>bufferSizeInBytes</i>	the length of the buffer in bytes

6.14.2.3 FTPixmapFont::~~FTPixmapFont ()

Destructor.

6.14.3 Member Function Documentation

6.14.3.1 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. 45).

Parameters

<i>slot</i>	A FreeType glyph slot.
-------------	------------------------

Returns

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

Implements **FTFont** (p. 43).

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

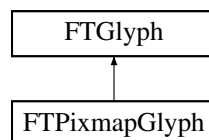
- **FTGLPixmapFont.h**

6.15 FTPixmapGlyph Class Reference

FTPixmapGlyph (p. 55) is a specialisation of **FTGlyph** (p. 45) for creating pixmaps.

```
#include <FTPixmapGlyph.h>
```

Inheritance diagram for FTPixmapGlyph:



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.

Additional Inherited Members

6.15.1 Detailed Description

FTPixmapGlyph (p. 55) is a specialisation of **FTGlyph** (p. 45) for creating pixmaps.

Definition at line 42 of file FTPixmapGlyph.h.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 FTPixmapGlyph::FTPixmapGlyph (FT_GlyphSlot *glyph*)

Constructor.

Parameters

<i>glyph</i>	The Freetype glyph to be processed
--------------	------------------------------------

6.15.2.2 virtual FTPixmapGlyph::~~FTPixmapGlyph () [virtual]

Destructor.

6.15.3 Member Function Documentation

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

Render this glyph at the current pen position.

Parameters

<i>pen</i>	The current pen position.
<i>renderMode</i>	Render mode to display

Returns

The advance distance for this glyph.

Implements **FTGlyph** (p. 47).

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

- **FTPixmapGlyph.h**

6.16 FTPoint Class Reference

FTPoint (p. 56) class is a basic 3-dimensional point or vector.

```
#include <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.

6.16.1 Detailed Description

FTPoint (p. 56) class is a basic 3-dimensional point or vector.

Definition at line 42 of file FTPoint.h.

6.16.2 Constructor & Destructor Documentation

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

Default constructor.

Point is set to zero.

Definition at line 48 of file `FTPoint.h`.

6.16.2.2 `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

<code>x</code>	First component
<code>y</code>	Second component
<code>z</code>	Third component

Definition at line 62 of file `FTPoint.h`.

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

Constructor.

This converts an `FT_Vector` to an **FTPoint** (p. 56)

Parameters

<code>ft_vector</code>	A freetype vector
------------------------	-------------------

Definition at line 75 of file `FTPoint.h`.

6.16.3 Member Function Documentation

6.16.3.1 `FTPoint FTPoint::Normalise ()`

Normalise a point's coordinates.

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

Returns

A vector of norm one.

6.16.3.2 `FTPoint::operator const FTGL_DOUBLE * () const` [inline]

Cast to `FTGL_DOUBLE*`.

Definition at line 240 of file `FTPoint.h`.

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

Operator `*` Scalar multiplication.

Parameters

<i>multiplier</i>

Returns

this multiplied by multiplier.

Definition at line 159 of file FTPoint.h.

6.16.3.4 FTPoint FTPoint::operator+ (const FTPoint & point) const [inline]

Operator +.

Parameters

<i>point</i>

Returns

this plus point.

Definition at line 112 of file FTPoint.h.

6.16.3.5 FTPoint& FTPoint::operator+= (const FTPoint & point) [inline]

Operator += In Place Addition.

Parameters

<i>point</i>

Returns

this plus point.

Definition at line 97 of file FTPoint.h.

6.16.3.6 FTPoint FTPoint::operator- (const FTPoint & point) const [inline]

Operator -.

Parameters

<i>point</i>

Returns

this minus point.

Definition at line 143 of file FTPoint.h.

6.16.3.7 FTPoint& FTPoint::operator-= (const FTPoint & point) [inline]

Operator -= In Place Substraction.

Parameters

<i>point</i>

Returns

this minus point.

Definition at line 128 of file FTPoint.h.

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

Operator ^ Vector product.

Parameters

<i>point</i>	Second point
--------------	--------------

Returns

this vector point.

Definition at line 204 of file FTPoint.h.

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

Setters.

Definition at line 249 of file FTPoint.h.

Referenced by FTBBox::operator|=().

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

Getters.

Definition at line 257 of file FTPoint.h.

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

Definition at line 260 of file FTPoint.h.

Referenced by FTFont::BBox().

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

Definition at line 250 of file FTPoint.h.

Referenced by FTBBox::operator|=().

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

Definition at line 258 of file FTPoint.h.

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

Definition at line 261 of file FTPoint.h.

Referenced by FTFont::BBox().

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

Definition at line 251 of file FTPoint.h.

Referenced by FTBBBox::operator|=().

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

Definition at line 259 of file FTPoint.h.

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

Definition at line 262 of file FTPoint.h.

Referenced by FTFont::BBox().

6.16.4 Friends And Related Function Documentation

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

Operator != Tests for non equality.

Parameters

<i>a</i>	
<i>b</i>	

Returns

true if a & b are not equal

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

Operator * Scalar multiplication.

Parameters

<i>point</i>	
<i>multiplier</i>	

Returns

multiplier multiplied by *point*.

Definition at line 177 of file FTPoint.h.

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

Operator * Scalar product.

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

a . *b* scalar product.

Definition at line 190 of file FTPoint.h.

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

Operator == Tests for equality.

Parameters

<i>a</i>	
<i>b</i>	

Returns

true if *a* & *b* are equal

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

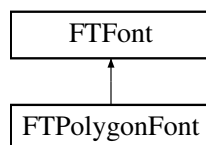
- **FTPoint.h**

6.17 FTPolygonFont Class Reference

FTPolygonFont (p. 62) is a specialisation of the **FTFont** (p. 36) class for handling tessellated Polygon Mesh fonts.

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

Inheritance diagram for FTPolygonFont:

**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.

6.17.1 Detailed Description

FTPolygonFont (p. 62) is a specialisation of the **FTFont** (p. 36) class for handling tessellated Polygon Mesh fonts.

See also

FTFont (p. 36)

Definition at line 45 of file FTGLPolygonFont.h.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 FTPolygonFont::FTPolygonFont (const char * *fontFilePath*)

Open and read a font file.

Sets Error flag.

Parameters

<i>fontFilePath</i>	font file path.
---------------------	-----------------

6.17.2.2 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. 19). The pointer must be valid while using **FTGL** (p. 19).

Parameters

<i>pBufferBytes</i>	the in-memory buffer
<i>bufferSizeInBytes</i>	the length of the buffer in bytes

6.17.2.3 FTPolygonFont::~~FTPolygonFont ()

Destructor.

6.17.3 Member Function Documentation

6.17.3.1 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. 45).

Parameters

<i>slot</i>	A FreeType glyph slot.
-------------	------------------------

Returns

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

Implements **FTFont** (p. 43).

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

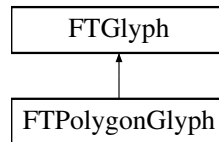
- **FTGLPolygonFont.h**

6.18 FTPolygonGlyph Class Reference

FTPolygonGlyph (p. 64) is a specialisation of **FTGlyph** (p. 45) for creating tessellated polygon glyphs.

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

Inheritance diagram for FTPolygonGlyph:



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.

Additional Inherited Members

6.18.1 Detailed Description

FTPolygonGlyph (p. 64) is a specialisation of **FTGlyph** (p. 45) for creating tessellated polygon glyphs.

Definition at line 43 of file FTPolyGlyph.h.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 FTPolygonGlyph::FTPolygonGlyph (FT_GlyphSlot glyph, float outset, bool useDisplayList)

Constructor.

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

Parameters

<i>glyph</i>	The Freetype glyph to be processed
<i>outset</i>	The outset distance
<i>useDisplayList</i>	Enable or disable the use of Display Lists for this glyph <code>true</code> turns ON display lists. <code>false</code> turns OFF display lists.

6.18.2.2 virtual FTPolygonGlyph::~~FTPolygonGlyph () [virtual]

Destructor.

6.18.3 Member Function Documentation

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

Render this glyph at the current pen position.

Parameters

<i>pen</i>	The current pen position.
<i>renderMode</i>	Render mode to display

Returns

The advance distance for this glyph.

Implements **FTGlyph** (p. 47).

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

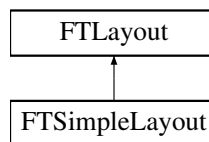
- **FTPolyGlyph.h**

6.19 FTSimpleLayout Class Reference

FTSimpleLayout (p. 66) is a specialisation of **FTLayout** (p. 48) for simple text boxes.

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

Inheritance diagram for FTSimpleLayout:



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

Additional Inherited Members

6.19.1 Detailed Description

FTSimpleLayout (p. 66) is a specialisation of **FTLayout** (p. 48) for simple text boxes.

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

See also

FTLayout (p. 48)

Definition at line 49 of file FTSimpleLayout.h.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 FTSimpleLayout::FTSimpleLayout ()

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

6.19.2.2 FTSimpleLayout::~~FTSimpleLayout ()

Destructor.

6.19.3 Member Function Documentation

6.19.3.1 virtual FTBBBox FTSimpleLayout::BBox (const char * *string*, const int *len* = -1, FTPoint *position* = FTPoint ()) [virtual]

Get the bounding box for a formatted string.

Parameters

<i>string</i>	A char string.
<i>len</i>	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
<i>position</i>	The pen position of the first character (optional).

Returns

The corresponding bounding box.

Implements **FTLayout** (p. 49).

6.19.3.2 virtual FTBBBox FTSimpleLayout::BBox (const wchar_t * *string*, const int *len* = -1, FTPoint *position* = FTPoint ()) [virtual]

Get the bounding box for a formatted string.

Parameters

<i>string</i>	A <code>wchar_t</code> string.
<i>len</i>	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
<i>position</i>	The pen position of the first character (optional).

Returns

The corresponding bounding box.

Implements **FTLayout** (p. 49).

6.19.3.3 **FTGL::TextAlignment** FTSimpleLayout::GetAlignment () const

Returns

The text alignment mode.

6.19.3.4 **FTFont*** FTSimpleLayout::GetFont ()

Returns

The current font.

6.19.3.5 **float** FTSimpleLayout::GetLineLength () const

Returns

The current line length.

6.19.3.6 **float** FTSimpleLayout::GetLineSpacing () const

Returns

The line spacing.

6.19.3.7 **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

<i>string</i>	'C' style string to be output.
<i>len</i>	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
<i>position</i>	The pen position of the first character (optional).
<i>renderMode</i>	Render mode to display (optional)

Implements **FTLayout** (p. 50).

6.19.3.8 **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

<i>string</i>	wchar_t string to be output.
<i>len</i>	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
<i>position</i>	The pen position of the first character (optional).
<i>renderMode</i>	Render mode to display (optional)

Implements **FTLayout** (p. 50).

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

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

Parameters

<i>Alignment</i>	The new alignment mode.
------------------	-------------------------

6.19.3.10 void FTSimpleLayout::SetFont (FTFont * *fontInit*)

Set the font to use for rendering the text.

Parameters

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

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

The maximum line length for formatting text.

Parameters

<i>LineLength</i>	The new line length.
-------------------	----------------------

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

Sets the line height.

Parameters

<i>LineSpacing</i>	The height of each line of text expressed as a percentage of the current fonts line height.
--------------------	---

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

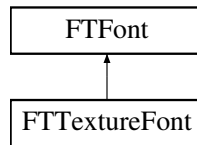
- **FTSimpleLayout.h**

6.20 FTTextureFont Class Reference

FTTextureFont (p. 69) is a specialisation of the **FTFont** (p. 36) class for handling Texture mapped fonts.

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

Inheritance diagram for FTTextureFont:



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.

6.20.1 Detailed Description

FTTextureFont (p. 69) is a specialisation of the **FTFont** (p. 36) class for handling Texture mapped fonts.

See also

FTFont (p. 36)

Definition at line 45 of file FTGLTextureFont.h.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 FTTextureFont::FTTextureFont (const char * fontFilePath)

Open and read a font file.

Sets Error flag.

Parameters

<i>fontFilePath</i>	font file path.
---------------------	-----------------

6.20.2.2 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. 19). The pointer must be valid while using **FTGL** (p. 19).

Parameters

<i>pBufferBytes</i>	the in-memory buffer
<i>bufferSizeInBytes</i>	the length of the buffer in bytes

6.20.2.3 virtual FTTextureFont::~FTTextureFont () [virtual]

Destructor.

6.20.3 Member Function Documentation

6.20.3.1 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. 45).

Parameters

<i>slot</i>	A FreeType glyph slot.
-------------	------------------------

Returns

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

Implements **FTFont** (p. 43).

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

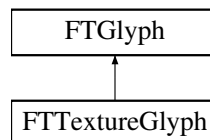
- **FTGLTextureFont.h**

6.21 FTTextureGlyph Class Reference

FTTextureGlyph (p. 71) is a specialisation of **FTGlyph** (p. 45) for creating texture glyphs.

```
#include <FTTextureGlyph.h>
```

Inheritance diagram for FTTextureGlyph:



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.

Additional Inherited Members

6.21.1 Detailed Description

FTTextureGlyph (p. 71) is a specialisation of **FTGlyph** (p. 45) for creating texture glyphs.

Definition at line 43 of file FTTextureGlyph.h.

6.21.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

6.21.2.2 virtual FTTextureGlyph::~FTTextureGlyph () [virtual]

Destructor.

6.21.3 Member Function Documentation

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

Render this glyph at the current pen position.

Parameters

<i>pen</i>	The current pen position.
<i>renderMode</i>	Render mode to display

Returns

The advance distance for this glyph.

Implements **FTGlyph** (p. 47).

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

- **FTTextureGlyph.h**

Chapter 7

File Documentation

7.1 faq.dox File Reference

7.2 FTBBox.h File Reference

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

Data Structures

- class **FTBBox**

FTBBox (p. 21) is a convenience class for handling bounding boxes.

7.3 FTBitmapGlyph.h File Reference

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

Data Structures

- class **FTBitmapGlyph**

FTBitmapGlyph (p. 25) is a specialisation of *FTGlyph* (p. 45) for creating bitmaps.

Functions

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

Create a specialisation of *FTGLglyph* for creating bitmaps.

7.3.1 Function Documentation

7.3.1.1 FTGLglyph* ftglCreateBitmapGlyph (FT_GlyphSlot glyph)

Create a specialisation of *FTGLglyph* for creating bitmaps.

Parameters

<i>glyph</i>	The Freetype glyph to be processed
--------------	------------------------------------

Returns

An FTGLglyph* object.

7.4 FTBuffer.h File Reference

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

Data Structures

- class **FTBuffer**
FTBuffer (p. 27) is a helper class for pixel buffers.

7.5 FTBufferFont.h File Reference

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

Data Structures

- class **FTBufferFont**
FTBufferFont (p. 30) is a specialisation of the *FTFont* (p. 36) class for handling memory buffer fonts.

Functions

- **FTGLfont * ftglCreateBufferFont** (const char *file)
Create a specialised FTGLfont object for handling memory buffer fonts.

7.5.1 Function Documentation

7.5.1.1 FTGLfont* ftglCreateBufferFont (const char * file)

Create a specialised FTGLfont object for handling memory buffer fonts.

Parameters

<i>file</i>	The font file name.
-------------	---------------------

Returns

An FTGLfont* object.

See also

FTGLfont (p. 77)

7.6 FTBufferGlyph.h File Reference

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

Data Structures

- class **FTBufferGlyph**

FTBufferGlyph (p. 32) is a specialisation of *FTGlyph* (p. 45) for memory buffer rendering.

7.7 FExtrdGlyph.h File Reference

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

Data Structures

- class **FExtrudeGlyph**

FExtrudeGlyph (p. 34) is a specialisation of *FTGlyph* (p. 45) for creating tessellated extruded polygon glyphs.

Macros

- #define **FExtrdGlyph FExtrudeGlyph**

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.

7.7.1 Macro Definition Documentation

7.7.1.1 #define FExtrdGlyph FExtrudeGlyph

Definition at line 77 of file FExtrdGlyph.h.

7.7.2 Function Documentation

7.7.2.1 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

<i>glyph</i>	The Freetype glyph to be processed
<i>depth</i>	The distance along the z axis to extrude the glyph

<i>frontOutset</i>	outset contour size
<i>backOutset</i>	outset contour size
<i>useDisplayList</i>	Enable or disable the use of Display Lists for this glyph <code>true</code> turns ON display lists. <code>false</code> turns OFF display lists.

Returns

An FTGLglyph* object.

7.8 FTFont.h File Reference

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

Data Structures

- class **FTFont**
FTFont (p. 36) is the public interface for the *FTGL* (p. 19) library.

Typedefs

- typedef struct _FTGLfont **FTGLfont**

Functions

- **FTGLfont * ftglCreateCustomFont** (char const *fontFilePath, void *data, **FTGLglyph** *(*makeglyph←
Callback)(FT_GlyphSlot, void *))
Create a custom FTGL (p. 19) font object.
- void **ftglDestroyFont** (**FTGLfont** *font)
Destroy an FTGL (p. 19) 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. 19).

- 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.

7.8.1 Typedef Documentation

7.8.1.1 typedef struct _FTGLfont FTGLfont

Definition at line 399 of file FTFont.h.

7.8.2 Function Documentation

7.8.2.1 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

<i>font</i>	An FTGLfont* object.
<i>data</i>	The in-memory buffer.
<i>size</i>	The length of the buffer in bytes.

Returns

1 if file has been attached successfully.

7.8.2.2 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

<i>font</i>	An FTGLfont* object.
-------------	----------------------

<i>path</i>	Auxilliary font file path.
-------------	----------------------------

Returns

1 if file has been attached successfully.

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

Create a custom **FTGL** (p. 19) font object.

Parameters

<i>fontFilePath</i>	The font file name.
<i>data</i>	A pointer to private data that will be passed to callbacks.
<i>makeglyph</i> ↔ <i>Callback</i>	A glyph-making callback function.

Returns

An FTGLfont* object.

7.8.2.4 void ftglDestroyFont (FTGLfont * *font*)

Destroy an **FTGL** (p. 19) font object.

Parameters

<i>font</i>	An FTGLfont* object.
-------------	----------------------

7.8.2.5 float ftglGetFontAdvance (FTGLfont * *font*, const char * *string*)

Get the advance width for a string.

Parameters

<i>font</i>	An FTGLfont* object.
<i>string</i>	A char string.

Returns

Advance width

7.8.2.6 float ftglGetFontAscender (FTGLfont * *font*)

Get the global ascender height for the face.

Parameters

<i>font</i>	An FTGLfont* object.
-------------	----------------------

Returns

Ascender height

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

Get the bounding box for a string.

Parameters

<i>font</i>	An FTGLfont* object.
<i>string</i>	A char buffer
<i>len</i>	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
<i>bounds</i>	An array of 6 float values where the bounding box's lower left near and upper right far 3D coordinates will be stored.

7.8.2.8 unsigned int ftglGetFontCharMapCount (FTGLfont * font)

Get the number of character maps in this face.

Parameters

<i>font</i>	An FTGLfont* object.
-------------	----------------------

Returns

character map count.

7.8.2.9 FT_Encoding* ftglGetFontCharMapList (FTGLfont * font)

Get a list of character maps in this face.

Parameters

<i>font</i>	An FTGLfont* object.
-------------	----------------------

Returns

pointer to the first encoding.

7.8.2.10 float ftglGetFontDescender (FTGLfont * font)

Gets the global descender height for the face.

Parameters

<i>font</i>	An FTGLfont* object.
-------------	----------------------

Returns

Descender height

7.8.2.11 FT_Error ftglGetFontError (FTGLfont * font)

Query a font for errors.

Parameters

<i>font</i>	An FTGLfont* object.
-------------	----------------------

Returns

The current error code.

7.8.2.12 unsigned int ftglGetFontSize (FTGLfont * *font*)

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

Parameters

<i>font</i>	An FTGLfont* object.
-------------	----------------------

Returns

face size

7.8.2.13 float ftglGetFontLineHeight (FTGLfont * font)

Gets the line spacing for the font.

Parameters

<i>font</i>	An FTGLfont* object.
-------------	----------------------

Returns

Line height

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

Render a string of characters.

Parameters

<i>font</i>	An FTGLfont* object.
<i>string</i>	Char string to be output.
<i>mode</i>	Render mode to display.

7.8.2.15 int ftglSetFontCharMap (FTGLfont * font, FT_Encoding encoding)

Set the character map for the face.

Parameters

<i>font</i>	An FTGLfont* object.
<i>encoding</i>	Freetype enumerate for char map code.

Returns

1 if charmap was valid and set correctly.

7.8.2.16 void ftglSetFontDepth (FTGLfont * font, float depth)

Set the extrusion distance for the font.

Only implemented by **FTExtrudeFont** (p. 33).

Parameters

<i>font</i>	An FTGLfont* object.
-------------	----------------------

<i>depth</i>	The extrusion distance.
--------------	-------------------------

7.8.2.17 void ftglSetFontDisplayList (FTGLfont * font, int useList)

Enable or disable the use of Display Lists inside **FTGL** (p. 19).

Parameters

<i>font</i>	An FTGLfont* object.
<i>useList</i>	1 turns ON display lists. 0 turns OFF display lists.

7.8.2.18 int ftglSetFontFaceSize (FTGLfont * font, unsigned int size, unsigned int res)

Set the char size for the current face.

Parameters

<i>font</i>	An FTGLfont* object.
<i>size</i>	The face size in points (1/72 inch).
<i>res</i>	The resolution of the target device, or 0 to use the default value of 72.

Returns

1 if size was set correctly.

7.8.2.19 void ftglSetFontOutset (FTGLfont * font, float front, float back)

Set the outset distance for the font.

Only **FTOutlineFont** (p. 51), **FTPolygonFont** (p. 62) and **FTExtrudeFont** (p. 33) implement front outset. Only **FT↔ExtrudeFont** (p. 33) implements back outset.

Parameters

<i>font</i>	An FTGLfont* object.
<i>front</i>	The front outset distance.
<i>back</i>	The back outset distance.

7.9 ftgl.dox File Reference

7.10 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

- **FTGL**

Macros

- **#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 }

7.10.1 Macro Definition Documentation

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

Definition at line 43 of file ftgl.h.

7.10.1.2 #define FTGL_END_C_DECLS }

Definition at line 44 of file ftgl.h.

7.10.1.3 #define FTGL_EXPORT

Definition at line 107 of file ftgl.h.

7.10.2 Typedef Documentation

7.10.2.1 typedef double FTGL_DOUBLE

Definition at line 38 of file ftgl.h.

7.10.2.2 typedef float FTGL_FLOAT

Definition at line 39 of file ftgl.h.

7.11 FTGLBitmapFont.h File Reference

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

Data Structures

- class **FTBitmapFont**

FTBitmapFont (p. 23) is a specialisation of the *FTFont* (p. 36) class for handling Bitmap fonts.

Macros

- #define **FTGLBitmapFont FTBitmapFont**

Functions

- **FTGLfont * ftglCreateBitmapFont** (const char *file)
Create a specialised FTGLfont object for handling bitmap fonts.

7.11.1 Macro Definition Documentation

7.11.1.1 #define FTGLBitmapFont FTBitmapFont

Definition at line 84 of file FTGLBitmapFont.h.

7.11.2 Function Documentation

7.11.2.1 FTGLfont* ftglCreateBitmapFont (const char * file)

Create a specialised FTGLfont object for handling bitmap fonts.

Parameters

<i>file</i>	The font file name.
-------------	---------------------

Returns

An FTGLfont* object.

See also

FTGLfont (p. 77)

7.12 FTGLExtrdFont.h File Reference

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

Data Structures

- class **FTE ExtrudeFont**

FTE ExtrudeFont (p. 33) is a specialisation of the *FTFont* (p. 36) class for handling extruded Polygon fonts.

Macros

- #define **FTGLExtrdFont FTE ExtrudeFont**

Functions

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

Create a specialised FTGLfont object for handling extruded poygon fonts.

7.12.1 Macro Definition Documentation

7.12.1.1 #define FTGLExtrdFont FTE ExtrudeFont

Definition at line 85 of file FTGLExtrdFont.h.

7.12.2 Function Documentation

7.12.2.1 FTGLfont* ftglCreateExtrudeFont (const char * file)

Create a specialised FTGLfont object for handling extruded poygon fonts.

Parameters

<i>file</i>	The font file name.
-------------	---------------------

Returns

An FTGLfont* object.

See also

FTGLfont (p. 77)

ftglCreatePolygonFont (p. 89)

7.13 FTGLOutlineFont.h File Reference

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

Data Structures

- class **FTOutlineFont**

FTOutlineFont (p. 51) is a specialisation of the *FTFont* (p. 36) class for handling Vector Outline fonts.

Macros

- #define **FTGLOutlineFont FTOutlineFont**

Functions

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

Create a specialised FTGLfont object for handling vector outline fonts.

7.13.1 Macro Definition Documentation

7.13.1.1 #define FTGLOutlineFont FTOutlineFont

Definition at line 84 of file FTGLOutlineFont.h.

7.13.2 Function Documentation

7.13.2.1 FTGLfont* ftglCreateOutlineFont (const char * file)

Create a specialised FTGLfont object for handling vector outline fonts.

Parameters

<i>file</i>	The font file name.
-------------	---------------------

Returns

An FTGLfont* object.

See also

FTGLfont (p. 77)

7.14 FTGLPixmapFont.h File Reference

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

Data Structures

- class **FTPixmapFont**

FTPixmapFont (p. 53) is a specialisation of the *FTFont* (p. 36) class for handling Pixmap (Grey Scale) fonts.

Macros

- `#define FTGLPixmapFont FTPixmapFont`

Functions

- **FTGLfont * ftgICreatePixmapFont** (const char *file)
Create a specialised FTGLfont object for handling pixmap (grey scale) fonts.

7.14.1 Macro Definition Documentation

7.14.1.1 #define FTGLPixmapFont FTPixmapFont

Definition at line 84 of file FTGLPixmapFont.h.

7.14.2 Function Documentation

7.14.2.1 FTGLfont* ftgICreatePixmapFont (const char * file)

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

Parameters

<i>file</i>	The font file name.
-------------	---------------------

Returns

An FTGLfont* object.

See also

FTGLfont (p. 77)

7.15 FTGLPolygonFont.h File Reference

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

Data Structures

- class **FTPolygonFont**
FTPolygonFont (p. 62) is a specialisation of the *FTFont* (p. 36) class for handling tessellated Polygon Mesh fonts.

Macros

- `#define FTGLPolygonFont FTPolygonFont`

Functions

- **FTGLfont * ftgICreatePolygonFont** (const char *file)
Create a specialised FTGLfont object for handling tessellated polygon mesh fonts.

7.15.1 Macro Definition Documentation

7.15.1.1 #define FTGLPolygonFont FTPolygonFont

Definition at line 84 of file FTGLPolygonFont.h.

7.15.2 Function Documentation

7.15.2.1 FTGLfont* ftglCreatePolygonFont (const char * file)

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

Parameters

<i>file</i>	The font file name.
-------------	---------------------

Returns

An FTGLfont* object.

See also

FTGLfont (p. 77)

7.16 FTGLTextureFont.h File Reference

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

Data Structures

- class **FTTextureFont**

FTTextureFont (p. 69) is a specialisation of the *FTFont* (p. 36) class for handling Texture mapped fonts.

Macros

- #define **FTGLTextureFont FTTextureFont**

Functions

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

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

7.16.1 Macro Definition Documentation

7.16.1.1 #define FTGLTextureFont FTTextureFont

Definition at line 84 of file FTGLTextureFont.h.

7.16.2 Function Documentation

7.16.2.1 FTGLfont* ftglCreateTextureFont (const char * *file*)

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

Parameters

<i>file</i>	The font file name.
-------------	---------------------

Returns

An FTGLfont* object.

See also

FTGLfont (p. 77)

7.17 FTGlyph.h File Reference

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

Data Structures

- class **FTGlyph**
FTGlyph (p. 45) is the base class for *FTGL* (p. 19) 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. 19) glyph object.
- void **ftglDestroyGlyph** (**FTGLglyph** *glyph)
Destroy an FTGL (p. 19) 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.

7.17.1 Typedef Documentation

7.17.1.1 typedef struct _FTGLglyph FTGLglyph

Definition at line 133 of file FTGlyph.h.

7.17.2 Function Documentation

7.17.2.1 **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. 19) glyph object.

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

Parameters

<i>base</i>	The base FTGLglyph* to subclass.
<i>data</i>	A pointer to private data that will be passed to callbacks.
<i>renderCallback</i>	A rendering callback function.
<i>destroyCallback</i>	A callback function to be called upon destruction.

Returns

An FTGLglyph* object.

7.17.2.2 `void ftglDestroyGlyph (FTGLglyph * glyph)`

Destroy an **FTGL** (p. 19) glyph object.

Parameters

<i>glyph</i>	An FTGLglyph* object.
--------------	-----------------------

7.17.2.3 `float ftglGetGlyphAdvance (FTGLglyph * glyph)`

Return the advance for a glyph.

Parameters

<i>glyph</i>	An FTGLglyph* object.
--------------	-----------------------

Returns

The advance's X component.

7.17.2.4 `void ftglGetGlyphBBox (FTGLglyph * glyph, float bounds[6])`

Return the bounding box for a glyph.

Parameters

<i>glyph</i>	An FTGLglyph* object.
<i>bounds</i>	An array of 6 float values where the bounding box's lower left near and upper right far 3D coordinates will be stored.

7.17.2.5 `FT_Error ftglGetGlyphError (FTGLglyph * glyph)`

Query a glyph for errors.

Parameters

<i>glyph</i>	An FTGLglyph* object.
--------------	-----------------------

Returns

The current error code.

7.17.2.6 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

<i>glyph</i>	An FTGLglyph* object.
<i>penx</i>	The current pen's X position.
<i>peny</i>	The current pen's Y position.
<i>renderMode</i>	Render mode to display
<i>advancex</i>	A pointer to an FTGL_DOUBLE where to write the advance's X component.
<i>advancey</i>	A pointer to an FTGL_DOUBLE where to write the advance's Y component.

7.18 FTLayout.h File Reference

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

Data Structures

- class **FTLayout**
FTLayout (p. 48) is the interface for layout managers that render text.

Typedefs

- typedef struct _FTGLlayout **FTGLlayout**

Functions

- void **ftglDestroyLayout** (FTGLlayout *layout)
Destroy an FTGL (p. 19) 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.

7.18.1 Typedef Documentation

7.18.1.1 typedef struct _FTGLlayout FTGLlayout

Definition at line 151 of file FTLayout.h.

7.18.2 Function Documentation

7.18.2.1 void ftglDestroyLayout (FTGLLayout * layout)

Destroy an **FTGL** (p. 19) layout object.

Parameters

<i>layout</i>	An FTGLLayout* object.
---------------	------------------------

7.18.2.2 void ftglGetLayoutBBox (FTGLLayout * layout, const char * string, float bounds[6])

Get the bounding box for a string.

Parameters

<i>layout</i>	An FTGLLayout* object.
<i>string</i>	A char buffer
<i>bounds</i>	An array of 6 float values where the bounding box's lower left near and upper right far 3D coordinates will be stored.

7.18.2.3 FT_Error ftglGetLayoutError (FTGLLayout * layout)

Query a layout for errors.

Parameters

<i>layout</i>	An FTGLLayout* object.
---------------	------------------------

Returns

The current error code.

7.18.2.4 void ftglRenderLayout (FTGLLayout * layout, const char * string, int mode)

Render a string of characters.

Parameters

<i>layout</i>	An FTGLLayout* object.
<i>string</i>	Char string to be output.
<i>mode</i>	Render mode to display.

7.19 FTOutlineGlyph.h File Reference

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

Data Structures

- class **FTOutlineGlyph**

FTOutlineGlyph (p. 52) is a specialisation of **FTGlyph** (p. 45) for creating outlines.

Functions

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

Create a specialisation of FTGLglyph for creating outlines.

7.19.1 Function Documentation

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

Create a specialisation of FTGLglyph for creating outlines.

Parameters

<i>glyph</i>	The Freetype glyph to be processed
<i>outset</i>	outset contour size
<i>useDisplayList</i>	Enable or disable the use of Display Lists for this glyph <code>true</code> turns ON display lists. <code>false</code> turns OFF display lists.

Returns

An FTGLglyph* object.

7.20 FTPixmapGlyph.h File Reference

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

Data Structures

- class **FTPixmapGlyph**

FTPixmapGlyph (p. 55) is a specialisation of *FTGlyph* (p. 45) for creating pixmaps.

Functions

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

Create a specialisation of FTGLglyph for creating pixmaps.

7.20.1 Function Documentation

7.20.1.1 FTGLglyph* ftglCreatePixmapGlyph (FT_GlyphSlot glyph)

Create a specialisation of FTGLglyph for creating pixmaps.

Parameters

<i>glyph</i>	The Freetype glyph to be processed
--------------	------------------------------------

Returns

An FTGLglyph* object.

7.21 FTPoint.h File Reference

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

Data Structures

- class **FTPoint**

FTPoint (p. 56) class is a basic 3-dimensional point or vector.

7.22 FTPolyGlyph.h File Reference

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

Data Structures

- class **FTPolygonGlyph**

FTPolygonGlyph (p. 64) is a specialisation of *FTGlyph* (p. 45) for creating tessellated polygon glyphs.

Macros

- `#define FTPolyGlyph FTPolygonGlyph`

Functions

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

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

7.22.1 Macro Definition Documentation

7.22.1.1 #define FTPolyGlyph FTPolygonGlyph

Definition at line 74 of file FTPolyGlyph.h.

7.22.2 Function Documentation

7.22.2.1 FTGLglyph* ftgCreatePolygonGlyph (FT_GlyphSlot glyph, float outset, int useDisplayList)

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

Parameters

<i>glyph</i>	The Freetype glyph to be processed
<i>outset</i>	outset contour size
<i>useDisplayList</i>	Enable or disable the use of Display Lists for this glyph <code>true</code> turns ON display lists. <code>false</code> turns OFF display lists.

Returns

An *FTGLglyph** object.

7.23 FTSimpleLayout.h File Reference

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

Data Structures

- class **FTSimpleLayout**

FTSimpleLayout (p. 66) is a specialisation of *FTLayout* (p. 48) 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 *)

7.23.1 Function Documentation

7.23.1.1 **FTGLlayout* ftglCreateSimpleLayout** (void)

7.23.1.2 **int ftglGetLayoutAlignement** (FTGLlayout *)

7.23.1.3 **FTGLfont* ftglGetLayoutFont** (FTGLlayout *)

7.23.1.4 **float ftglGetLayoutLineLength** (FTGLlayout *)

7.23.1.5 **float ftglGetLayoutLineSpacing** (FTGLlayout *)

7.23.1.6 **void ftglSetLayoutAlignment** (FTGLlayout *, const int)

7.23.1.7 **void ftglSetLayoutFont** (FTGLlayout *, FTGLfont *)

7.23.1.8 **void ftglSetLayoutLineLength** (FTGLlayout *, const float)

7.23.1.9 **void ftglSetLayoutLineSpacing** (FTGLlayout *, const float)

7.24 FTTextureGlyph.h File Reference

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

Data Structures

- class **FTTextureGlyph**

FTTextureGlyph (p. 71) is a specialisation of *FTGlyph* (p. 45) 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.

7.24.1 Function Documentation

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

Create a specialisation of FTGLglyph for creating pixmaps.

Parameters

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

Returns

An FTGLglyph* object.

7.25 projects_using_ftgl.txt File Reference

7.26 tutorial.dox File Reference

Index

- ~FTBBox
 - FTBBox, 22
- ~FTBitmapFont
 - FTBitmapFont, 24
- ~FTBitmapGlyph
 - FTBitmapGlyph, 27
- ~FTBuffer
 - FTBuffer, 28
- ~FTBufferFont
 - FTBufferFont, 31
- ~FTBufferGlyph
 - FTBufferGlyph, 32
- ~FTEXtrudeFont
 - FTEXtrudeFont, 34
- ~FTEXtrudeGlyph
 - FTEXtrudeGlyph, 35
- ~FTFont
 - FTFont, 38
- ~FTGlyph
 - FTGlyph, 46
- ~FTLayout
 - FTLayout, 49
- ~FTOutlineFont
 - FTOutlineFont, 52
- ~FTOutlineGlyph
 - FTOutlineGlyph, 53
- ~FTPixmapFont
 - FTPixmapFont, 55
- ~FTPixmapGlyph
 - FTPixmapGlyph, 56
- ~FTPolygonFont
 - FTPolygonFont, 63
- ~FTPolygonGlyph
 - FTPolygonGlyph, 64
- ~FTSimpleLayout
 - FTSimpleLayout, 67
- ~FTTextureFont
 - FTTextureFont, 71
- ~FTTextureGlyph
 - FTTextureGlyph, 72
- ALIGN_CENTER
 - FTGL, 19
- ALIGN_JUSTIFY
 - FTGL, 19
- ALIGN_LEFT
 - FTGL, 19
- ALIGN_RIGHT
 - FTGL, 19
- Advance
 - FTFont, 38, 39
 - FTGlyph, 46
- Ascender
 - FTFont, 39
- Attach
 - FTFont, 39
- BBox
 - FTFont, 40, 41
 - FTGlyph, 46
 - FTLayout, 49
 - FTSimpleLayout, 67
- CharMap
 - FTFont, 41
- CharMapCount
 - FTFont, 41
- CharMapList
 - FTFont, 41
- Depth
 - FTFont, 41
- Descender
 - FTFont, 42
- Error
 - FTFont, 42
 - FTGlyph, 47
 - FTLayout, 50
- FTBBox, 21
 - ~FTBBox, 22
 - FTBBox, 22
 - Invalidate, 22
 - IsValid, 22
 - Lower, 22
 - operator+=, 23
 - operator|=, 23
 - SetDepth, 23
 - Upper, 23
- FTBBox.h, 73
- FTBitmapFont, 23
 - ~FTBitmapFont, 24
 - FTBitmapFont, 24
 - FTFont, 44
 - MakeGlyph, 25
- FTBitmapGlyph, 25
 - ~FTBitmapGlyph, 27
 - FTBitmapGlyph, 26
 - FTGlyph, 47
 - Render, 27

- FTBitmapGlyph.h, 73
 - ftglCreateBitmapGlyph, 73
- FTBuffer, 27
 - ~FTBuffer, 28
 - FTBuffer, 28
 - Height, 28
 - Pixels, 28
 - Pos, 28
 - Size, 30
 - Width, 30
- FTBuffer.h, 74
- FTBufferFont, 30
 - ~FTBufferFont, 31
 - FTBufferFont, 31
 - FTFont, 44
 - MakeGlyph, 31
- FTBufferFont.h, 74
 - ftglCreateBufferFont, 74
- FTBufferGlyph, 32
 - ~FTBufferGlyph, 32
 - FTBufferGlyph, 32
 - FTGlyph, 47
 - Render, 32
- FTBufferGlyph.h, 75
- FTEextrdGlyph
 - FTEextrdGlyph.h, 75
- FTEextrdGlyph.h, 75
 - FTEextrdGlyph, 75
 - ftglCreateExtrudeGlyph, 75
- FTEextrudeFont, 33
 - ~FTEextrudeFont, 34
 - FTEextrudeFont, 34
 - FTFont, 44
 - MakeGlyph, 34
- FTEextrudeGlyph, 34
 - ~FTEextrudeGlyph, 35
 - FTEextrudeGlyph, 35
 - FTGlyph, 47
 - Render, 35
- FTFont, 36
 - ~FTFont, 38
 - Advance, 38, 39
 - Ascender, 39
 - Attach, 39
 - BBox, 40, 41
 - CharMap, 41
 - CharMapCount, 41
 - CharMapList, 41
 - Depth, 41
 - Descender, 42
 - Error, 42
 - FTBitmapFont, 44
 - FTBufferFont, 44
 - FTEextrudeFont, 44
 - FTFont, 38
 - FTFontImpl, 44
 - FTOutlineFont, 45
 - FTPixmapFont, 45
 - FTPolygonFont, 45
 - FTTextureFont, 45
 - FaceSize, 42
 - GlyphLoadFlags, 42
 - LineHeight, 43
 - MakeGlyph, 43
 - Outset, 43
 - Render, 43, 44
 - UseDisplayList, 44
- FTFont.h, 76
 - FTGLfont, 77
 - ftglAttachData, 77
 - ftglAttachFile, 77
 - ftglCreateCustomFont, 78
 - ftglDestroyFont, 78
 - ftglGetFontAdvance, 78
 - ftglGetFontAscender, 78
 - ftglGetFontBBox, 78
 - ftglGetFontCharMapCount, 80
 - ftglGetFontCharMapList, 80
 - ftglGetFontDescender, 80
 - ftglGetFontError, 80
 - ftglGetFontFaceSize, 80
 - ftglGetFontLineHeight, 82
 - ftglRenderFont, 82
 - ftglSetFontCharMap, 82
 - ftglSetFontDepth, 82
 - ftglSetFontDisplayList, 83
 - ftglSetFontFaceSize, 83
 - ftglSetFontOutset, 83
- FTFontImpl
 - FTFont, 44
- FTGL, 19
 - ALIGN_CENTER, 19
 - ALIGN_JUSTIFY, 19
 - ALIGN_LEFT, 19
 - ALIGN_RIGHT, 19
 - RENDER_ALL, 19
 - RENDER_BACK, 19
 - RENDER_FRONT, 19
 - RENDER_SIDE, 19
 - RenderMode, 19
 - TextAlignment, 19
- FTGL_BEGIN_C_DECLS
 - ftgl.h, 84
- FTGL_DOUBLE
 - ftgl.h, 85
- FTGL_END_C_DECLS
 - ftgl.h, 84
- FTGL_EXPORT
 - ftgl.h, 85
- FTGL_FLOAT
 - ftgl.h, 85
- FTGLBitmapFont
 - FTGLBitmapFont.h, 85
- FTGLBitmapFont.h, 85
 - FTGLBitmapFont, 85
 - ftglCreateBitmapFont, 85

- FTGLExtrdFont
 - FTGLExtrdFont.h, 86
- FTGLExtrdFont.h, 86
 - FTGLExtrdFont, 86
 - ftglCreateExtrudeFont, 86
- FTGLOutlineFont
 - FTGLOutlineFont.h, 87
- FTGLOutlineFont.h, 87
 - FTGLOutlineFont, 87
 - ftglCreateOutlineFont, 87
- FTGLPixmapFont
 - FTGLPixmapFont.h, 88
- FTGLPixmapFont.h, 87
 - FTGLPixmapFont, 88
 - ftglCreatePixmapFont, 88
- FTGLPolygonFont
 - FTGLPolygonFont.h, 89
- FTGLPolygonFont.h, 88
 - FTGLPolygonFont, 89
 - ftglCreatePolygonFont, 89
- FTGLTextureFont
 - FTGLTextureFont.h, 89
- FTGLTextureFont.h, 89
 - FTGLTextureFont, 89
 - ftglCreateTextureFont, 90
- FTGLfont
 - FTFont.h, 77
- FTGLglyph
 - FTGlyph.h, 91
- FTGLlayout
 - FTLayout.h, 93
- FTGlyph, 45
 - ~FTGlyph, 46
 - Advance, 46
 - BBox, 46
 - Error, 47
 - FTBitmapGlyph, 47
 - FTBufferGlyph, 47
 - FTExtrudeGlyph, 47
 - FTGlyph, 46
 - FTOutlineGlyph, 47
 - FTPixmapGlyph, 47
 - FTPolygonGlyph, 47
 - FTTextureGlyph, 48
 - Render, 47
- FTGlyph.h, 91
 - FTGLglyph, 91
 - ftglCreateCustomGlyph, 92
 - ftglDestroyGlyph, 92
 - ftglGetGlyphAdvance, 92
 - ftglGetGlyphBBox, 92
 - ftglGetGlyphError, 92
 - ftglRenderGlyph, 93
- FTLayout, 48
 - ~FTLayout, 49
 - BBox, 49
 - Error, 50
 - FTLayout, 49
 - FTSimpleLayout, 50
 - Render, 50
- FTLayout.h, 93
 - FTGLlayout, 93
 - ftglDestroyLayout, 94
 - ftglGetLayoutBBox, 94
 - ftglGetLayoutError, 94
 - ftglRenderLayout, 94
- FTOutlineFont, 51
 - ~FTOutlineFont, 52
 - FTFont, 45
 - FTOutlineFont, 51
 - MakeGlyph, 52
- FTOutlineGlyph, 52
 - ~FTOutlineGlyph, 53
 - FTGlyph, 47
 - FTOutlineGlyph, 53
 - Render, 53
- FTOutlineGlyph.h, 94
 - ftglCreateOutlineGlyph, 95
- FTPixmapFont, 53
 - ~FTPixmapFont, 55
 - FTFont, 45
 - FTPixmapFont, 54
 - MakeGlyph, 55
- FTPixmapGlyph, 55
 - ~FTPixmapGlyph, 56
 - FTGlyph, 47
 - FTPixmapGlyph, 56
 - Render, 56
- FTPixmapGlyph.h, 95
 - ftglCreatePixmapGlyph, 95
- FTPPoint, 56
 - FTPPoint, 58
 - Normalise, 58
 - operator const FTGL_DOUBLE *, 58
 - operator!=, 61
 - operator*, 58, 61
 - operator^, 60
 - operator+, 59
 - operator+=", 59
 - operator-, 59
 - operator-=, 59
 - operator==, 62
 - X, 60
 - Xf, 60
 - Y, 60
 - Yf, 60
 - Z, 60, 61
 - Zf, 61
- FTPPoint.h, 96
- FTPolyGlyph
 - FTPolyGlyph.h, 96
- FTPolyGlyph.h, 96
 - FTPolyGlyph, 96
 - ftglCreatePolygonGlyph, 96
- FTPolygonFont, 62
 - ~FTPolygonFont, 63

- FTFont, 45
- FTPolygonFont, 63
- MakeGlyph, 63
- FTPolygonGlyph, 64
 - ~FTPolygonGlyph, 64
 - FTGlyph, 47
 - FTPolygonGlyph, 64
 - Render, 64
- FTSimpleLayout, 66
 - ~FTSimpleLayout, 67
 - BBox, 67
 - FTLayout, 50
 - FTSimpleLayout, 67
 - GetAlignment, 68
 - GetFont, 68
 - GetLineLength, 68
 - GetLineSpacing, 68
 - Render, 68
 - SetAlignment, 69
 - SetFont, 69
 - SetLineLength, 69
 - SetLineSpacing, 69
- FTSimpleLayout.h, 97
 - ftglCreateSimpleLayout, 97
 - ftglGetLayoutAlignment, 97
 - ftglGetLayoutFont, 97
 - ftglGetLayoutLineLength, 97
 - ftglGetLayoutLineSpacing, 97
 - ftglSetLayoutAlignment, 97
 - ftglSetLayoutFont, 97
 - ftglSetLayoutLineLength, 97
 - ftglSetLayoutLineSpacing, 97
- FTTextureFont, 69
 - ~FTTextureFont, 71
 - FTFont, 45
 - FTTextureFont, 70
 - MakeGlyph, 71
- FTTextureGlyph, 71
 - ~FTTextureGlyph, 72
 - FTGlyph, 48
 - FTTextureGlyph, 72
 - Render, 72
- FTTextureGlyph.h, 97
 - ftglCreateTextureGlyph, 98
- FaceSize
 - FTFont, 42
- faq.dox, 73
- ftgl.dox, 83
- ftgl.h, 83
 - FTGL_BEGIN_C_DECLS, 84
 - FTGL_DOUBLE, 85
 - FTGL_END_C_DECLS, 84
 - FTGL_EXPORT, 85
 - FTGL_FLOAT, 85
- ftglAttachData
 - FTFont.h, 77
- ftglAttachFile
 - FTFont.h, 77
- ftglCreateBitmapFont
 - FTGLBitmapFont.h, 85
- ftglCreateBitmapGlyph
 - FTBitmapGlyph.h, 73
- ftglCreateBufferFont
 - FTBufferFont.h, 74
- ftglCreateCustomFont
 - FTFont.h, 78
- ftglCreateCustomGlyph
 - FTGlyph.h, 92
- ftglCreateExtrudeFont
 - FTGLExtrdFont.h, 86
- ftglCreateExtrudeGlyph
 - FTExtrdGlyph.h, 75
- ftglCreateOutlineFont
 - FTGLOutlineFont.h, 87
- ftglCreateOutlineGlyph
 - FTOutlineGlyph.h, 95
- ftglCreatePixmapFont
 - FTGLPixmapFont.h, 88
- ftglCreatePixmapGlyph
 - FTPixmapGlyph.h, 95
- ftglCreatePolygonFont
 - FTGLPolygonFont.h, 89
- ftglCreatePolygonGlyph
 - FTPolyGlyph.h, 96
- ftglCreateSimpleLayout
 - FTSimpleLayout.h, 97
- ftglCreateTextureFont
 - FTGLTextureFont.h, 90
- ftglCreateTextureGlyph
 - FTTextureGlyph.h, 98
- ftglDestroyFont
 - FTFont.h, 78
- ftglDestroyGlyph
 - FTGlyph.h, 92
- ftglDestroyLayout
 - FTLayout.h, 94
- ftglGetFontAdvance
 - FTFont.h, 78
- ftglGetFontAscender
 - FTFont.h, 78
- ftglGetFontBBox
 - FTFont.h, 78
- ftglGetFontCharMapCount
 - FTFont.h, 80
- ftglGetFontCharMapList
 - FTFont.h, 80
- ftglGetFontDescender
 - FTFont.h, 80
- ftglGetFontError
 - FTFont.h, 80
- ftglGetFontFaceSize
 - FTFont.h, 80
- ftglGetFontLineHeight
 - FTFont.h, 82
- ftglGetGlyphAdvance
 - FTGlyph.h, 92

- ftglGetGlyphBBox
 - FTGlyph.h, 92
- ftglGetGlyphError
 - FTGlyph.h, 92
- ftglGetLayoutAlignment
 - FTSimpleLayout.h, 97
- ftglGetLayoutBBox
 - FTLayout.h, 94
- ftglGetLayoutError
 - FTLayout.h, 94
- ftglGetLayoutFont
 - FTSimpleLayout.h, 97
- ftglGetLayoutLineLength
 - FTSimpleLayout.h, 97
- ftglGetLayoutLineSpacing
 - FTSimpleLayout.h, 97
- ftglRenderFont
 - FTFont.h, 82
- ftglRenderGlyph
 - FTGlyph.h, 93
- ftglRenderLayout
 - FTLayout.h, 94
- ftglSetFontCharMap
 - FTFont.h, 82
- ftglSetFontDepth
 - FTFont.h, 82
- ftglSetFontDisplayList
 - FTFont.h, 83
- ftglSetFontFaceSize
 - FTFont.h, 83
- ftglSetFontOutset
 - FTFont.h, 83
- ftglSetLayoutAlignment
 - FTSimpleLayout.h, 97
- ftglSetLayoutFont
 - FTSimpleLayout.h, 97
- ftglSetLayoutLineLength
 - FTSimpleLayout.h, 97
- ftglSetLayoutLineSpacing
 - FTSimpleLayout.h, 97

- GetAlignment
 - FTSimpleLayout, 68
- GetFont
 - FTSimpleLayout, 68
- GetLineLength
 - FTSimpleLayout, 68
- GetLineSpacing
 - FTSimpleLayout, 68
- GlyphLoadFlags
 - FTFont, 42

- Height
 - FTBuffer, 28

- Invalidate
 - FTBBox, 22
- IsValid
 - FTBBox, 22

- LineHeight
 - FTFont, 43
- Lower
 - FTBBox, 22

- MakeGlyph
 - FTBitmapFont, 25
 - FTBufferFont, 31
 - FTExtrudeFont, 34
 - FTFont, 43
 - FTOutlineFont, 52
 - FTPixmapFont, 55
 - FTPolygonFont, 63
 - FTTextureFont, 71

- Normalise
 - FTPoint, 58

- operator const FTGL_DOUBLE *
 - FTPoint, 58
- operator!=
 - FTPoint, 61
- operator*
 - FTPoint, 58, 61
- operator^
 - FTPoint, 60
- operator+
 - FTPoint, 59
- operator+=
 - FTBBox, 23
 - FTPoint, 59
- operator-
 - FTPoint, 59
- operator-=
 - FTPoint, 59
- operator==
 - FTPoint, 62
- operator |=
 - FTBBox, 23
- Outset
 - FTFont, 43

- Pixels
 - FTBuffer, 28
- Pos
 - FTBuffer, 28
- projects_using_ftgl.txt, 98

- RENDER_ALL
 - FTGL, 19
- RENDER_BACK
 - FTGL, 19
- RENDER_FRONT
 - FTGL, 19
- RENDER_SIDE
 - FTGL, 19
- Render
 - FTBitmapGlyph, 27
 - FTBufferGlyph, 32

- FTExtrudeGlyph, 35
- FTFont, 43, 44
- FTGlyph, 47
- FTLayout, 50
- FTOutlineGlyph, 53
- FTPixmapGlyph, 56
- FTPolygonGlyph, 64
- FTSimpleLayout, 68
- FTTextureGlyph, 72
- RenderMode
 - FTGL, 19
- SetAlignment
 - FTSimpleLayout, 69
- SetDepth
 - FTBox, 23
- SetFont
 - FTSimpleLayout, 69
- SetLineLength
 - FTSimpleLayout, 69
- SetLineSpacing
 - FTSimpleLayout, 69
- Size
 - FTBuffer, 30
- TextAlignment
 - FTGL, 19
- tutorial.dox, 98
- Upper
 - FTBox, 23
- UseDisplayList
 - FTFont, 44
- Width
 - FTBuffer, 30
- X
 - FTPoint, 60
- Xf
 - FTPoint, 60
- Y
 - FTPoint, 60
- Yf
 - FTPoint, 60
- Z
 - FTPoint, 60, 61
- Zf
 - FTPoint, 61