

FTGL

2.1.3~rc5

Generated by Doxygen 1.7.6.1

Sat Feb 25 2012 14:09:22

pt3em2

pt3em

pt3em

pt3em

pt3em

pt3em

Generated on Sat Feb 25 2012 14:09:19 for FTGL by Doxygen

pt3em

pt3em

pt3em

pt3em

pt3emi

pt3em

pt3emii

pt3em

pt3em

pt3em

pt3em

pt3em

Generated on Sat Feb 25 2012 14:09:19 for FTGL by Doxygen

Contents

1 FTGL User Guide	3
1.1 Introduction	3
1.2 Documentation	3
1.3 Additional information	4
2 Frequently Asked Questions	5
2.1 FAQ	5
2.1.1 When I try to compile %FTGL it complains about a missing file from the include: #include <ft2build.h>	5
2.1.2 Is it possible to map a font to a "unit" size? My application re- lies on the fonts being a certain "physical" height (in OpenGL coordinate space) rather than a point size in display space. Any thoughts/suggestions?	5
3 Projects using FTGL	7
3.1 %FTGL language bindings	7
3.1.1 %FTGL#	7
3.1.2 GIGuiA	7
3.1.3 Ruby %FTGL	7
3.1.4 PyFTGL	7
3.2 Projects currently using %FTGL	8
3.2.1 Agent World	8
3.2.2 Amaltheia	8
3.2.3 Armagetron Advanced	8
3.2.4 Audicle	8
3.2.5 Battlestar T.U.X.	8
3.2.6 BJS	9

pt3em	
3.2.7 Blender	9
3.2.8 Breve	9
3.2.9 BZFlag	9
3.2.10 Capture The Flag	9
3.2.11 Cello	9
3.2.12 Chimera	10
3.2.13 Cinepaint	10
3.2.14 Duel	10
3.2.15 Empty Clip	10
3.2.16 Freebox	10
3.2.17 Gem	10
3.2.18 GLMayan	10
3.2.19 Glover	11
3.2.20 Ivf++	11
3.2.21 Jahshaka	11
3.2.22 Karaoke FX	11
3.2.23 Libinstrudeo	11
3.2.24 Light Speed!	11
3.2.25 MySQL GUI Tools	12
3.2.26 OctPlot	12
3.2.27 Open ActiveWrl	12
3.2.28 OpenEaagles	12
3.2.29 OpenGC	12
3.2.30 OpenSG	12
3.2.31 Panthera	13
3.2.32 Planet Penguin Racer	13
3.2.33 projectM	13
3.2.34 Puzzle Bobble 3D	13
3.2.35 ROOT	13
3.2.36 SCIRun	13
3.2.37 TINE	13
3.2.38 Tiny Planet	14
3.2.39 Truevision	14

pt3em	
3.2.40 Tulip	14
3.2.41 Ubit	14
3.2.42 VRS	14
3.2.43 VTK	14
3.2.44 XLock	14
3.3 Projects that used to use %FTGL	15
3.3.1 GNU Backgammon	15
3.3.2 OpenSceneGraph	15
3.3.3 Teddy	15
3.3.4 VigiPac	15
4 FTGL tutorial	17
4.1 Starting to use %FTGL	17
4.2 Choosing a font type	17
4.2.1 Raster fonts	17
4.2.2 Vector fonts	17
4.2.3 Textured fonts	18
4.3 Create font objects	18
4.3.1 in C	18
4.3.2 in C++	19
4.4 More font commands	19
4.4.1 Font metrics	19
4.4.2 Specifying a character map encoding	20
4.5 Sample font manager class	21
5 Namespace Documentation	23
5.1 FTGL Namespace Reference	23
5.1.1 Enumeration Type Documentation	23
5.1.1.1 RenderMode	23
5.1.1.2 TextAlignment	23
6 Data Structure Documentation	25
6.1 FTBBox Class Reference	25
6.1.1 Detailed Description	26

	pt3em	
6.1.2	Constructor & Destructor Documentation	26
6.1.2.1	FTBBox	26
6.1.2.2	FTBBox	26
6.1.2.3	FTBBox	26
6.1.2.4	FTBBox	26
6.1.2.5	~FTBBox	26
6.1.3	Member Function Documentation	27
6.1.3.1	Invalidate	27
6.1.3.2	IsValid	27
6.1.3.3	Lower	27
6.1.3.4	operator+=	27
6.1.3.5	operator =	27
6.1.3.6	SetDepth	28
6.1.3.7	Upper	28
6.2	FTBitmapFont Class Reference	28
6.2.1	Detailed Description	29
6.2.2	Constructor & Destructor Documentation	29
6.2.2.1	FTBitmapFont	29
6.2.2.2	FTBitmapFont	29
6.2.2.3	~FTBitmapFont	30
6.2.3	Member Function Documentation	30
6.2.3.1	MakeGlyph	30
6.3	FTBitmapGlyph Class Reference	30
6.3.1	Detailed Description	31
6.3.2	Constructor & Destructor Documentation	31
6.3.2.1	FTBitmapGlyph	31
6.3.2.2	~FTBitmapGlyph	31
6.3.3	Member Function Documentation	31
6.3.3.1	Render	31
6.4	FTBuffer Class Reference	32
6.4.1	Detailed Description	32
6.4.2	Constructor & Destructor Documentation	33
6.4.2.1	FTBuffer	33

pt3em		
6.4.2.2	~FTBuffer	33
6.4.3	Member Function Documentation	33
6.4.3.1	Height	33
6.4.3.2	Pixels	33
6.4.3.3	Pos	33
6.4.3.4	Pos	34
6.4.3.5	Size	34
6.4.3.6	Width	34
6.5	FTBufferFont Class Reference	34
6.5.1	Detailed Description	35
6.5.2	Constructor & Destructor Documentation	35
6.5.2.1	FTBufferFont	35
6.5.2.2	FTBufferFont	36
6.5.2.3	~FTBufferFont	36
6.5.3	Member Function Documentation	36
6.5.3.1	MakeGlyph	36
6.6	FTBufferGlyph Class Reference	37
6.6.1	Detailed Description	37
6.6.2	Constructor & Destructor Documentation	37
6.6.2.1	FTBufferGlyph	37
6.6.2.2	~FTBufferGlyph	38
6.6.3	Member Function Documentation	38
6.6.3.1	Render	38
6.7	FTExtrudeFont Class Reference	38
6.7.1	Detailed Description	39
6.7.2	Constructor & Destructor Documentation	39
6.7.2.1	FTExtrudeFont	39
6.7.2.2	FTExtrudeFont	40
6.7.2.3	~FTExtrudeFont	40
6.7.3	Member Function Documentation	40
6.7.3.1	MakeGlyph	40
6.8	FTExtrudeGlyph Class Reference	41
6.8.1	Detailed Description	41

pt3em		
6.8.2	Constructor & Destructor Documentation	41
6.8.2.1	FTEXtrudeGlyph	41
6.8.2.2	~FTEXtrudeGlyph	42
6.8.3	Member Function Documentation	42
6.8.3.1	Render	42
6.9	FTFont Class Reference	42
6.9.1	Detailed Description	45
6.9.2	Constructor & Destructor Documentation	45
6.9.2.1	FTFont	45
6.9.2.2	FTFont	45
6.9.2.3	~FTFont	46
6.9.3	Member Function Documentation	46
6.9.3.1	Advance	46
6.9.3.2	Advance	46
6.9.3.3	Ascender	47
6.9.3.4	Attach	47
6.9.3.5	Attach	47
6.9.3.6	BBox	48
6.9.3.7	BBox	48
6.9.3.8	BBox	49
6.9.3.9	BBox	49
6.9.3.10	CharMap	49
6.9.3.11	CharMapCount	50
6.9.3.12	CharMapList	50
6.9.3.13	Depth	50
6.9.3.14	Descender	50
6.9.3.15	Error	51
6.9.3.16	FaceSize	51
6.9.3.17	FaceSize	51
6.9.3.18	GlyphLoadFlags	51
6.9.3.19	LineHeight	52
6.9.3.20	MakeGlyph	52
6.9.3.21	Outset	52

pt3em	
6.9.3.22	Outset 53
6.9.3.23	Render 53
6.9.3.24	Render 53
6.9.3.25	UseDisplayList 54
6.9.4	Friends And Related Function Documentation 54
6.9.4.1	FTBitmapFont 54
6.9.4.2	FTBufferFont 54
6.9.4.3	FTExtrudeFont 54
6.9.4.4	FTFontImpl 54
6.9.4.5	FTOutlineFont 54
6.9.4.6	FTPixmapFont 55
6.9.4.7	FTPolygonFont 55
6.9.4.8	FTTextureFont 55
6.10	FTGlyph Class Reference 55
6.10.1	Detailed Description 56
6.10.2	Constructor & Destructor Documentation 56
6.10.2.1	FTGlyph 56
6.10.2.2	~FTGlyph 57
6.10.3	Member Function Documentation 57
6.10.3.1	Advance 57
6.10.3.2	BBox 57
6.10.3.3	Error 57
6.10.3.4	Render 57
6.10.4	Friends And Related Function Documentation 58
6.10.4.1	FTBitmapGlyph 58
6.10.4.2	FTBufferGlyph 58
6.10.4.3	FTExtrudeGlyph 58
6.10.4.4	FTOutlineGlyph 58
6.10.4.5	FTPixmapGlyph 58
6.10.4.6	FTPolygonGlyph 58
6.10.4.7	FTTextureGlyph 58
6.11	FTLayout Class Reference 59
6.11.1	Detailed Description 60

pt3em	
6.11.2	Constructor & Destructor Documentation 60
6.11.2.1	FTLayout 60
6.11.2.2	~FTLayout 60
6.11.3	Member Function Documentation 60
6.11.3.1	BBox 60
6.11.3.2	BBox 61
6.11.3.3	Error 61
6.11.3.4	Render 61
6.11.3.5	Render 62
6.11.4	Friends And Related Function Documentation 62
6.11.4.1	FTSimpleLayout 62
6.12	FTOutlineFont Class Reference 62
6.12.1	Detailed Description 63
6.12.2	Constructor & Destructor Documentation 63
6.12.2.1	FTOutlineFont 63
6.12.2.2	FTOutlineFont 63
6.12.2.3	~FTOutlineFont 64
6.12.3	Member Function Documentation 64
6.12.3.1	MakeGlyph 64
6.13	FTOutlineGlyph Class Reference 64
6.13.1	Detailed Description 65
6.13.2	Constructor & Destructor Documentation 65
6.13.2.1	FTOutlineGlyph 65
6.13.2.2	~FTOutlineGlyph 65
6.13.3	Member Function Documentation 66
6.13.3.1	Render 66
6.14	FTPixmapFont Class Reference 66
6.14.1	Detailed Description 67
6.14.2	Constructor & Destructor Documentation 67
6.14.2.1	FTPixmapFont 67
6.14.2.2	FTPixmapFont 67
6.14.2.3	~FTPixmapFont 68
6.14.3	Member Function Documentation 68

pt3em	
6.14.3.1	MakeGlyph 68
6.15	FTPixmapGlyph Class Reference 68
6.15.1	Detailed Description 69
6.15.2	Constructor & Destructor Documentation 69
6.15.2.1	FTPixmapGlyph 69
6.15.2.2	~FTPixmapGlyph 69
6.15.3	Member Function Documentation 69
6.15.3.1	Render 69
6.16	FTPoint Class Reference 70
6.16.1	Detailed Description 71
6.16.2	Constructor & Destructor Documentation 71
6.16.2.1	FTPoint 71
6.16.2.2	FTPoint 71
6.16.2.3	FTPoint 72
6.16.3	Member Function Documentation 72
6.16.3.1	Normalise 72
6.16.3.2	operator const FTGL_DOUBLE * 72
6.16.3.3	operator* 72
6.16.3.4	operator+ 73
6.16.3.5	operator+= 73
6.16.3.6	operator- 73
6.16.3.7	operator-= 74
6.16.3.8	operator^ 74
6.16.3.9	X 74
6.16.3.10	X 75
6.16.3.11	Xf 75
6.16.3.12	Y 75
6.16.3.13	Y 75
6.16.3.14	Yf 75
6.16.3.15	Z 75
6.16.3.16	Z 75
6.16.3.17	Zf 75
6.16.4	Friends And Related Function Documentation 76

pt3em	
6.16.4.1 operator!=	76
6.16.4.2 operator*	76
6.16.4.3 operator*	76
6.16.4.4 operator==	77
6.17 FTPolygonFont Class Reference	77
6.17.1 Detailed Description	78
6.17.2 Constructor & Destructor Documentation	78
6.17.2.1 FTPolygonFont	78
6.17.2.2 FTPolygonFont	78
6.17.2.3 ~FTPolygonFont	79
6.17.3 Member Function Documentation	79
6.17.3.1 MakeGlyph	79
6.18 FTPolygonGlyph Class Reference	79
6.18.1 Detailed Description	80
6.18.2 Constructor & Destructor Documentation	80
6.18.2.1 FTPolygonGlyph	80
6.18.2.2 ~FTPolygonGlyph	80
6.18.3 Member Function Documentation	81
6.18.3.1 Render	81
6.19 FTSimpleLayout Class Reference	81
6.19.1 Detailed Description	82
6.19.2 Constructor & Destructor Documentation	82
6.19.2.1 FTSimpleLayout	82
6.19.2.2 ~FTSimpleLayout	83
6.19.3 Member Function Documentation	83
6.19.3.1 BBox	83
6.19.3.2 BBox	83
6.19.3.3 GetAlignment	84
6.19.3.4 GetFont	84
6.19.3.5 GetLineLength	84
6.19.3.6 GetLineSpacing	84
6.19.3.7 Render	84
6.19.3.8 Render	85

pt3em	
6.19.3.9	SetAlignment 85
6.19.3.10	SetFont 85
6.19.3.11	SetLineLength 85
6.19.3.12	SetLineSpacing 86
6.20	FTTextureFont Class Reference 86
6.20.1	Detailed Description 87
6.20.2	Constructor & Destructor Documentation 87
6.20.2.1	FTTextureFont 87
6.20.2.2	FTTextureFont 87
6.20.2.3	~FTTextureFont 88
6.20.3	Member Function Documentation 88
6.20.3.1	MakeGlyph 88
6.21	FTTextureGlyph Class Reference 88
6.21.1	Detailed Description 89
6.21.2	Constructor & Destructor Documentation 89
6.21.2.1	FTTextureGlyph 89
6.21.2.2	~FTTextureGlyph 89
6.21.3	Member Function Documentation 89
6.21.3.1	Render 90
7	File Documentation 91
7.1	faq.dox File Reference 91
7.2	FTBBox.h File Reference 91
7.3	FTBitmapGlyph.h File Reference 91
7.3.1	Function Documentation 92
7.3.1.1	ftglCreateBitmapGlyph 92
7.4	FTBuffer.h File Reference 92
7.5	FTBufferFont.h File Reference 92
7.5.1	Function Documentation 93
7.5.1.1	ftglCreateBufferFont 93
7.6	FTBufferGlyph.h File Reference 93
7.7	FTEtrdGlyph.h File Reference 93
7.7.1	Define Documentation 94

pt3em		
7.7.1.1	FTExtrdGlyph	94
7.7.2	Function Documentation	94
7.7.2.1	ftglCreateExtrudeGlyph	94
7.8	FTFont.h File Reference	94
7.8.1	Typedef Documentation	96
7.8.1.1	FTGLfont	96
7.8.2	Function Documentation	96
7.8.2.1	ftglAttachData	96
7.8.2.2	ftglAttachFile	96
7.8.2.3	ftglCreateCustomFont	97
7.8.2.4	ftglDestroyFont	97
7.8.2.5	ftglGetFontAdvance	97
7.8.2.6	ftglGetFontAscender	98
7.8.2.7	ftglGetFontBBox	98
7.8.2.8	ftglGetFontCharMapCount	98
7.8.2.9	ftglGetFontCharMapList	99
7.8.2.10	ftglGetFontDescender	99
7.8.2.11	ftglGetFontError	99
7.8.2.12	ftglGetFontFaceSize	99
7.8.2.13	ftglGetFontLineHeight	100
7.8.2.14	ftglRenderFont	100
7.8.2.15	ftglSetFontCharMap	100
7.8.2.16	ftglSetFontDepth	101
7.8.2.17	ftglSetFontDisplayList	101
7.8.2.18	ftglSetFontFaceSize	101
7.8.2.19	ftglSetFontOutset	102
7.9	ftgl.dox File Reference	102
7.10	ftgl.h File Reference	102
7.10.1	Define Documentation	103
7.10.1.1	FTGL_BEGIN_C_DECLS	103
7.10.1.2	FTGL_END_C_DECLS	103
7.10.1.3	FTGL_EXPORT	103
7.10.2	Typedef Documentation	103

pt3em	
7.10.2.1 FTGL_DOUBLE	103
7.10.2.2 FTGL_FLOAT	103
7.11 FTGLBitmapFont.h File Reference	103
7.11.1 Define Documentation	104
7.11.1.1 FTGLBitmapFont	104
7.11.2 Function Documentation	104
7.11.2.1 ftglCreateBitmapFont	104
7.12 FTGLExtrdFont.h File Reference	105
7.12.1 Define Documentation	105
7.12.1.1 FTGLExtrdFont	105
7.12.2 Function Documentation	105
7.12.2.1 ftglCreateExtrudeFont	105
7.13 FTGLOutlineFont.h File Reference	106
7.13.1 Define Documentation	106
7.13.1.1 FTGLOutlineFont	106
7.13.2 Function Documentation	106
7.13.2.1 ftglCreateOutlineFont	106
7.14 FTGLPixmapFont.h File Reference	107
7.14.1 Define Documentation	107
7.14.1.1 FTGLPixmapFont	107
7.14.2 Function Documentation	108
7.14.2.1 ftglCreatePixmapFont	108
7.15 FTGLPolygonFont.h File Reference	108
7.15.1 Define Documentation	108
7.15.1.1 FTGLPolygonFont	108
7.15.2 Function Documentation	109
7.15.2.1 ftglCreatePolygonFont	109
7.16 FTGLTextureFont.h File Reference	109
7.16.1 Define Documentation	109
7.16.1.1 FTGLTextureFont	109
7.16.2 Function Documentation	110
7.16.2.1 ftglCreateTextureFont	110
7.17 FTGlyph.h File Reference	110

pt3em	
7.17.1	Typedef Documentation 111
7.17.1.1	FTGLglyph 111
7.17.2	Function Documentation 111
7.17.2.1	ftglCreateCustomGlyph 111
7.17.2.2	ftglDestroyGlyph 112
7.17.2.3	ftglGetGlyphAdvance 112
7.17.2.4	ftglGetGlyphBBox 112
7.17.2.5	ftglGetGlyphError 112
7.17.2.6	ftglRenderGlyph 113
7.18	FTLayout.h File Reference 113
7.18.1	Typedef Documentation 114
7.18.1.1	FTGLLayout 114
7.18.2	Function Documentation 114
7.18.2.1	ftglDestroyLayout 114
7.18.2.2	ftglGetLayoutBBox 114
7.18.2.3	ftglGetLayoutError 114
7.18.2.4	ftglRenderLayout 115
7.19	FTOutlineGlyph.h File Reference 115
7.19.1	Function Documentation 115
7.19.1.1	ftglCreateOutlineGlyph 115
7.20	FTPixmapGlyph.h File Reference 116
7.20.1	Function Documentation 116
7.20.1.1	ftglCreatePixmapGlyph 116
7.21	FTPoint.h File Reference 117
7.22	FTPolyGlyph.h File Reference 117
7.22.1	Define Documentation 117
7.22.1.1	FTPolyGlyph 117
7.22.2	Function Documentation 118
7.22.2.1	ftglCreatePolygonGlyph 118
7.23	FTSimpleLayout.h File Reference 118
7.23.1	Function Documentation 119
7.23.1.1	ftglCreateSimpleLayout 119
7.23.1.2	ftglGetLayoutAlignement 119

pt3em	
7.23.1.3	ftglGetLayoutFont 119
7.23.1.4	ftglGetLayoutLineLength 119
7.23.1.5	ftglGetLayoutLineSpacing 119
7.23.1.6	ftglSetLayoutAlignment 119
7.23.1.7	ftglSetLayoutFont 119
7.23.1.8	ftglSetLayoutLineLength 119
7.23.1.9	ftglSetLayoutLineSpacing 119
7.24	FTTextureGlyph.h File Reference 119
7.24.1	Function Documentation 119
7.24.1.1	ftglCreateTextureGlyph 119
7.25	projects_using_ftgl.txt File Reference 120
7.26	tutorial.dox File Reference 120

pt3em

pt3em

pt3em2

pt3em

pt3em**CONTENTS**

pt3em

pt3em

pt3em

Generated on Sat Feb 25 2012 14:09:19 for FTGL by Doxygen

pt3em

pt3em

pt3em

pt3em

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. 17)

pt3em

pt3em

pt3em

pt3em

- C API reference:
 - **FTGlyph.h** (p. 110)
 - **FTFont.h** (p. 94)
 - **FTLayout.h** (p. 113)
- C++ API reference:
 - class **FTGlyph** (p. 55)
 - class **FTFont** (p. 42)
 - class **FTLayout** (p. 59)

1.3 Additional information

- **Frequently Asked Questions** (p. 5)
- **Projects using FTGL** (p. 7)

pt3em

pt3em

pt3em

pt3em

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 gl-Scale but I understand if you don't want to.

pt3em

pt3em

pt3em

pt3em

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 problem, 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.

pt3em

pt3em

pt3em

pt3em

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, relaying (almost) only on OpenGL. Hence should be rather platform-independant.

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.

pt3em

pt3em

pt3em

pt3em

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.

pt3em

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.

pt3em

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.

pt3em

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.

pt3em

3.2.25 MySQL GUI Tools

MySQL GUI Tools (<http://dev.mysql.com/downloads/gui-tools/5.0.0.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 a 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.

pt3em

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 3-D 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.

pt3em

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 screen-saver and screen locking utility with additional OpenGL and XPM modes.

pt3em

3.3 Projects that used to use %FTGL

3.3.1 GNU Backgammon

GNU Backgammon (<http://www.gnu.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.

pt3em16

pt3em

pt3emProjects using FTGL

pt3em

pt3em

pt3em16 generated on Sat Feb 25 2012 14:09:19 for FTGL by Doxygen

pt3em

pt3em

pt3em

pt3em

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.

pt3em

pt3em

pt3em

pt3em

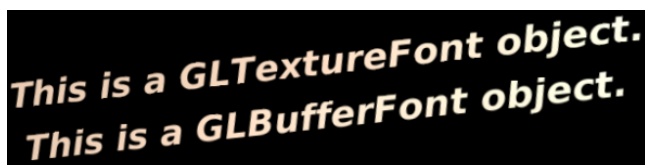
- Polygon fonts use planar triangle meshes and can be texture-mapped.
- Outline fonts use OpenGL lines.
- Extruded fonts are extruded polygon fonts, with the front, back and side meshes renderable separately to apply different effects and materials.



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;
```

```

pt3em

/* 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;

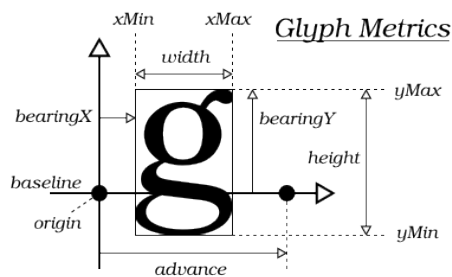
// Set the font size and render a small text.
font.FaceSize(72);
font.Render("Hello World!");

```

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 **FTFont::Ascender()** (p. 47), **FTFont::Descender()** (p. 50) and **FTFont::Advance()** (p. 46) an approximate bounding box can be calculated.

For an exact bounding box, use the **FTFont::BBox()** (p. 48) function. This function returns the extent of the volume containing 'string'. 0.0 on the y axis will be aligned with the font baseline.

pt3em

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 **FTFont::CharMap()** (p. 49).

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())
```

```

pt3em
    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();
    }
}

```

```
pt3em
    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;
};
```

pt3em

pt3em

pt3em

pt3em

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

pt3emEnumerator:

RENDER_FRONT
RENDER_BACK
RENDER_SIDE
RENDER_ALL

Definition at line 53 of file ftgl.h.

5.1.1.2 enum FTGL::TextAlignment

pt3emEnumerator:

ALIGN_LEFT
ALIGN_CENTER
ALIGN_RIGHT
ALIGN_JUSTIFY

pt3em

pt3em

pt3em

pt3em

Definition at line 61 of file ftgl.h.

Chapter 6

Data Structure Documentation

6.1 FTBBox Class Reference

FTBBox (p. 25) 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

pt3em**6.1.1 Detailed Description**

FTBBox (p. 25) 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 FTBBox::FTBBox()** [inline]

Default constructor.

Bounding box is set to zero.

Definition at line 48 of file FTBBox.h.

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

Constructor.

Definition at line 56 of file FTBBox.h.

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

Constructor.

Definition at line 64 of file FTBBox.h.

6.1.2.4 FTBBox::FTBBox(FT_GlyphSlot *glyph*) [inline]

Constructor.

Extracts a bounding box from a freetype glyph. Uses the control box for the glyph.

FT_Glyph_Get_CBox()

pt3emParameters

pt3emglyph	pt3emA freetype glyph
------------	-----------------------

Definition at line 75 of file FTBBox.h.

6.1.2.5 FTBBox::~~FTBBox() [inline]

Destructor.

pt3em

Definition at line 93 of file FTBBox.h.

6.1.3 Member Function Documentation

6.1.3.1 void FTBBox::Invalidate () [inline]

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

Definition at line 100 of file FTBBox.h.

6.1.3.2 bool FTBBox::IsValid () [inline]

Determines if this bounding box is valid.

pt3emReturns

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

Definition at line 112 of file FTBBox.h.

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

Definition at line 165 of file FTBBox.h.

Referenced by FTFont::BBox().

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

Move the Bounding Box by a vector.

pt3emParameters

pt3em <i>vector</i>	pt3em The vector to move the bbox in 3D space.
---------------------	--

Definition at line 124 of file FTBBox.h.

6.1.3.5 FTBBox& FTBBox::operator|=(const FTBBox & *bbbox*) [inline]

Combine two bounding boxes.

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

pt3em

pt3emParameters

pt3em	pt3em
-------	-------

pt3emThe bounding box to merge with the second one.

Definition at line 138 of file FTBBox.h.

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

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

Definition at line 150 of file FTBBox.h.

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

Definition at line 159 of file FTBBox.h.

Referenced by FTFont::BBox().

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

- FTBBox.h

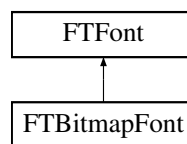
6.2 FTBitmapFont Class Reference

FTBitmapFont (p. 28) is a specialisation of the **FTFont** (p. 42) class for handling Bitmap fonts.

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

Inheritance diagram for FTBitmapFont:

pt3em



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.

pt3em

- `~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. 28) is a specialisation of the **FTFont** (p. 42) class for handling Bitmap fonts.

pt3em See also

FTFont (p. 42)

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.

pt3em Parameters

pt3em	pt3em	pt3emfont file path.
	<i>fontFilePath</i>	

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. 23). The pointer must be valid while using **FTGL** (p. 23).

pt3em Parameters

pt3em	pt3em	pt3emthe in-memory buffer
	<i>pBufferBytes</i>	

pt3em

pt3embuffer-SizeInBytes	pt3emthe 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. 55).

pt3emParameters

pt3emslot	pt3emA FreeType glyph slot.
-----------	-----------------------------

pt3emReturns

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

Implements **FTFont** (p. 52).

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

- **FTGLBitmapFont.h**

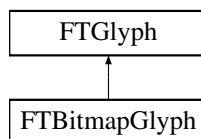
6.3 FTBitmapGlyph Class Reference

FTBitmapGlyph (p. 30) is a specialisation of **FTGlyph** (p. 55) for creating bitmaps.

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

Inheritance diagram for FTBitmapGlyph:

pt3em



pt3em

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.

6.3.1 Detailed Description

FTBitmapGlyph (p. 30) is a specialisation of **FTGlyph** (p. 55) 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.

pt3emParameters

pt3em glyph	pt3em 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.

pt3emParameters

pt3em pen	pt3em The current pen position.
pt3em renderMode	pt3em Render mode to display

pt3em

pt3emReturns

The advance distance for this glyph.

Implements **FTGlyph** (p. 57).

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

- **FTBitmapGlyph.h**

6.4 FTBuffer Class Reference

FTBuffer (p. 32) 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. 32) is a helper class for pixel buffers.

It provides the interface between **FTBufferFont** (p. 34) and **FTBufferGlyph** (p. 37) to optimise rendering operations.

pt3em
pt3emSee also

FTBufferGlyph (p. 37)
FTBufferFont (p. 34)

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.

pt3emReturns

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.

pt3emReturns

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.

pt3em

pt3emReturns

The pen's position as an **FTPoint** (p. 70) 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.

pt3emParameters

pt3em	pt3em	pt3em	An FTPoint (p. 70) 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.

pt3emParameters

pt3em	pt3em	pt3em	The buffer's desired width, in pixels.
	pt3em	pt3em	The buffer's desired height, in pixels.

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

Get the buffer's width.

pt3emReturns

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

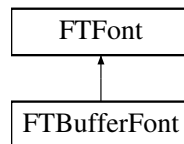
FTBufferFont (p. 34) is a specialisation of the **FTFont** (p. 42) class for handling memory buffer fonts.

pt3em

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

Inheritance diagram for FTBufferFont:

pt3em



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. 34) is a specialisation of the **FTFont** (p. 42) class for handling memory buffer fonts.

pt3emSee also

FTFont (p. 42)

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.

pt3em

pt3emParameters

pt3em pt3em <i>fontFilePath</i>	pt3emfont 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. 23). The pointer must be valid while using **FTGL** (p. 23).

pt3emParameters

pt3em pt3em <i>pBufferBytes</i>	pt3emthe in-memory buffer
pt3em <i>buffer-SizeInBytes</i>	pt3emthe 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 *s/ot*) [protected, virtual]

Construct a glyph of the correct type.

Clients must override the function and return their specialised **FTGlyph** (p. 55).

pt3emParameters

pt3empt3em <i>s/ot</i>	pt3emA FreeType glyph slot.
------------------------	-----------------------------

pt3emReturns

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

Implements **FTFont** (p. 52).

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

pt3em

- **FTBufferFont.h**

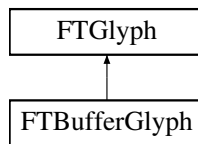
6.6 FTBufferGlyph Class Reference

FTBufferGlyph (p. 37) is a specialisation of **FTGlyph** (p. 55) for memory buffer rendering.

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

Inheritance diagram for FTBufferGlyph:

pt3em



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.

6.6.1 Detailed Description

FTBufferGlyph (p. 37) is a specialisation of **FTGlyph** (p. 55) 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.

pt3emParameters

pt3emglyph	pt3emThe FreeType glyph to be processed
pt3embuffer	pt3emAn FTBuffer (p. 32) object in which to render the glyph.

pt3em

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.

pt3emParameters

pt3em <i>pen</i>	pt3em The current pen position.
pt3em <i>renderMode</i>	pt3em Render mode to display

pt3emReturns

The advance distance for this glyph.

Implements **FTGlyph** (p. 57).

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

- **FTBufferGlyph.h**

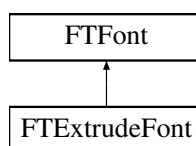
6.7 FTExtrudeFont Class Reference

FTExtrudeFont (p. 38) is a specialisation of the **FTFont** (p. 42) class for handling extruded Polygon fonts.

```
#include <FTGLeXtrdFont.h>
```

Inheritance diagram for FTExtrudeFont:

pt3em



pt3em

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. 38) is a specialisation of the **FTFont** (p. 42) class for handling extruded Polygon fonts.

pt3emSee also

- FTFont** (p. 42)
- FTPolygonFont** (p. 77)

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.

pt3emParameters

pt3em	pt3em	pt3emfont file path.
<i>fontFilePath</i>		

pt3em

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. 23). The pointer must be valid while using **FTGL** (p. 23).

pt3emParameters

pt3em <code>pBufferBytes</code>	pt3emthe in-memory buffer
pt3em <code>bufferSizeInBytes</code>	pt3emthe 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. 55).

pt3emParameters

pt3em <code>slot</code>	pt3emA FreeType glyph slot.
-------------------------	-----------------------------

pt3emReturns

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

Implements **FTFont** (p. 52).

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

- **FTGLExtrdFont.h**

pt3em

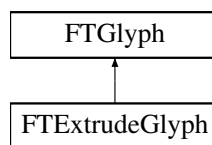
6.8 FTEXtrudeGlyph Class Reference

FTEXtrudeGlyph (p. 41) is a specialisation of **FTGlyph** (p. 55) for creating tessellated extruded polygon glyphs.

```
#include <FTEXtrdGlyph.h>
```

Inheritance diagram for FTEXtrudeGlyph:

pt3em



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.

6.8.1 Detailed Description

FTEXtrudeGlyph (p. 41) is a specialisation of **FTGlyph** (p. 55) 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.

pt3em

pt3emParameters

pt3emglyph	pt3emThe FreeType glyph to be processed
pt3emdepth	pt3emThe distance along the z axis to extrude the glyph
pt3emfrontOutset	pt3emoutset contour size
pt3embackOutset	pt3emoutset contour size
pt3emuseDisplayList	pt3emEnable 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.

pt3emParameters

pt3empen	pt3emThe current pen position.
pt3emrenderMode	pt3emRender mode to display

pt3emReturns

The advance distance for this glyph.

Implements **FTGlyph** (p. 57).

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

- **FTExtrdGlyph.h**

6.9 FTFont Class Reference

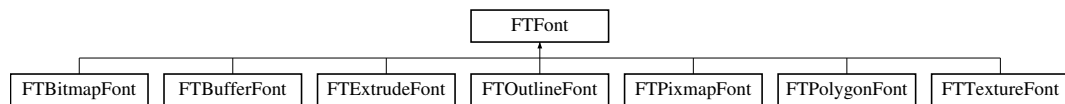
FTFont (p. 42) is the public interface for the **FTGL** (p. 23) library.

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

pt3em

Inheritance diagram for FTFont:

pt3em



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. 23).
- 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

pt3em

Gets the line spacing for the font.

- virtual **FTBBBox 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 &l1x, float &l1y, float &l2x, float &l2y, float &urx, float &ury, float &urz)

Get the bounding box for a string (deprecated).

- virtual **FTBBBox 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 &l1x, float &l1y, float &l2x, float &l2y, 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**

pt3em

- class **FTOutlineFont**
- class **FTPixmapFont**
- class **FTPolygonFont**
- class **FTTextureFont**
- class **FTFontImpl**

6.9.1 Detailed Description

FTFont (p. 42) is the public interface for the **FTGL** (p. 23) 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. 51). Check the freetype file `fterrdef.h` for error definitions.

pt3emSee 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.

pt3emParameters

pt3em	pt3em	pt3emfont file path.
	<i>fontFilePath</i>	

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. 23). The pointer must be valid while using **FTGL** (p. 23).

pt3em

pt3emParameters

pt3em pt3em <i>pBufferBytes</i>	pt3emthe in-memory buffer
pt3em <i>buffer-SizeInBytes</i>	pt3emthe 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.

pt3emParameters

pt3em <i>string</i>	pt3em'C' style string to be checked.
pt3em <i>len</i>	pt3emThe length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
pt3em <i>spacing</i>	pt3emA displacement vector to add after each character has been checked (optional).

pt3emReturns

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.

pt3emParameters

pt3em <i>string</i>	pt3emA wchar_t string
pt3em <i>len</i>	pt3emThe length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
pt3em <i>spacing</i>	pt3emA displacement vector to add after each character has been checked (optional).

pt3em

pt3emReturns

The string's advance width.

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

Get the global ascender height for the face.

pt3emReturns

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.

pt3emParameters

pt3em	pt3em	pt3emauxilliary font file path.
	<i>fontFilePath</i>	

pt3emReturns

`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.

pt3emParameters

pt3em	pt3em	pt3emthe in-memory buffer.
	<i>pBufferBytes</i>	
pt3em	<i>buffer-SizeInBytes</i>	pt3emthe length of the buffer in bytes.

pt3em

pt3emReturns

`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.

pt3emParameters

pt3em <i>string</i>	pt3emA char buffer.
pt3em <i>len</i>	pt3emThe length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
pt3em <i>position</i>	pt3emThe pen position of the first character (optional).
pt3em <i>spacing</i>	pt3emA displacement vector to add after each character has been checked (optional).

pt3emReturns

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).

pt3emParameters

pt3em <i>string</i>	pt3emA char buffer.
pt3em <i>llx</i>	pt3emLower left near x coordinate.
pt3em <i>lly</i>	pt3emLower left near y coordinate.
pt3em <i>llz</i>	pt3emLower left near z coordinate.
pt3em <i>urx</i>	pt3emUpper right far x coordinate.
pt3em <i>ury</i>	pt3emUpper right far y coordinate.
pt3em <i>urz</i>	pt3emUpper 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().

pt3em

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.

pt3emParameters

pt3em	pt3em
pt3em	pt3em
pt3em	pt3em
pt3em	pt3em

pt3emReturns

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).

pt3emParameters

pt3em	pt3em
pt3em	pt3em
pt3em	pt3em
pt3em	pt3em
pt3em	pt3em
pt3em	pt3em
pt3em	pt3em

Definition at line 286 of file FTFont.h.

References BBox(), FTBBBox::Lower(), FTBBBox::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.

pt3em

pt3emParameters

pt3em <i>encoding</i>	pt3emFreetype enumerate for char map code.
--------------------------	--

pt3emReturns

`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.

pt3emReturns

character map count.

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

Get a list of character maps in this face.

pt3emReturns

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. 38)

pt3emParameters

pt3em <i>depth</i>	pt3emThe extrusion distance.
-----------------------	------------------------------

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

Gets the global descender height for the face.

pt3em

pt3emReturns

Descender height

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

Queries the Font for errors.

pt3emReturns

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.

pt3emParameters

pt3emsize	pt3emthe face size in points (1/72 inch)
pt3emres	pt3emthe resolution of the target device.

pt3emReturns

`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).

pt3emReturns

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.

pt3em

pt3emParameters

pt3em	<i>flags</i>	pt3em	The glyph loading flags.
-------	--------------	-------	--------------------------

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

Gets the line spacing for the font.

pt3emReturns

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. 55).

pt3emParameters

pt3em	<i>slot</i>	pt3em	A FreeType glyph slot.
-------	-------------	-------	------------------------

pt3emReturns

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

Implemented in **FTExtrudeFont** (p. 40), **FTBitmapFont** (p. 30), **FTOutlineFont** (p. 64), **FTPixmapFont** (p. 68), **FTPolygonFont** (p. 79), **FTTextureFont** (p. 88), and **FTBufferFont** (p. 36).

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

Set the outset distance for the font.

Only implemented by **FTOutlineFont** (p. 62), **FTPolygonFont** (p. 77) and **FTExtrudeFont** (p. 38)

pt3emParameters

pt3em	<i>outset</i>	pt3em	The outset distance.
-------	---------------	-------	----------------------

Parameters

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. 38)

Parameters

front	The front outset distance.
back	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

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

Returns

The new pen position after the last character was output.

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

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

pt3em

pt3em <i>position</i>	pt3emThe pen position of the first character (optional).
pt3em <i>spacing</i>	pt3emA displacement vector to add after each character has been displayed (optional).
pt3em <i>renderMode</i>	pt3emRender mode to use for display (optional).

pt3emReturns

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. 23).

pt3emParameters

pt3em <i>useList</i>	pt3emtrue 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.

pt3em

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**

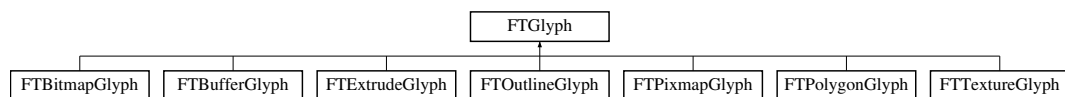
6.10 FTGlyph Class Reference

FTGlyph (p. 55) is the base class for **FTGL** (p. 23) glyphs.

#include <FTGlyph.h>

Inheritance diagram for FTGlyph:

pt3em



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.

pt3em

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. 55) is the base class for **FTGL** (p. 23) 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.

pt3emSee also

- FTBBox** (p. 25)
- FTPoint** (p. 70)

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.

pt3emParameters

pt3emglyph	pt3emThe Freetype glyph to be processed
------------	---

pt3em

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.

pt3emReturns

advance width.

6.10.3.2 virtual const FTBBBox& FTGlyph::BBox () const [virtual]

Return the bounding box for this glyph.

pt3emReturns

bounding box.

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

Queries for errors.

pt3emReturns

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.

pt3emParameters

pt3em pen	pt3emThe current pen position.
pt3em renderMode	pt3emRender mode to display

pt3em

pt3emReturns

The advance distance for this glyph.

Implemented in **FTExtrudeGlyph** (p. 42), **FTTextureGlyph** (p. 90), **FTPolygonGlyph** (p. 81), **FTOutlineGlyph** (p. 66), **FTBitmapGlyph** (p. 31), **FTPixmapGlyph** (p. 69), and **FTBufferGlyph** (p. 38).

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**

pt3em

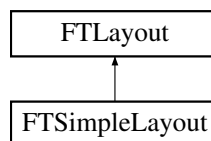
6.11 FTLayout Class Reference

FTLayout (p. 59) is the interface for layout managers that render text.

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

Inheritance diagram for FTLayout:

pt3em



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`

pt3em

6.11.1 Detailed Description

FTLayout (p. 59) 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.

pt3emSee also

FTFont (p. 42)

FTBBox (p. 25)

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.

pt3emParameters

pt3em <code>string</code>	pt3emA char string.
pt3em <code>len</code>	pt3emThe length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
pt3em <code>position</code>	pt3emThe pen position of the first character (optional).

pt3emReturns

The corresponding bounding box.

Implemented in **FTSimpleLayout** (p. 83).

6.11.3.2

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

Get the bounding box for a formatted string.

Parameters

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

Returns

The corresponding bounding box.

Implemented in **FTSimpleLayout** (p. 83).

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

Queries the Layout for errors.

Returns

The current error code.

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

Render a string of characters.

Parameters

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

Implemented in **FTSimpleLayout** (p. 84).

pt3em

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

Render a string of characters.

pt3emParameters

<code>pt3em <i>string</i></code>	pt3emwchar_t string to be output.
<code>pt3em <i>len</i></code>	pt3emThe length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
<code>pt3em <i>position</i></code>	pt3emThe pen position of the first character (optional).
<code>pt3em <i>renderMode</i></code>	pt3emRender mode to display (optional)

Implemented in **FTSimpleLayout** (p. 85).

6.11.4 Friends And Related Function Documentation

6.11.4.1 `friend class FTSimpleLayout` [`friend`]

Definition at line 67 of file FTLayout.h.

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

- **FTLayout.h**

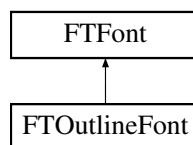
6.12 FTOutlineFont Class Reference

FTOutlineFont (p. 62) is a specialisation of the **FTFont** (p. 42) class for handling Vector Outline fonts.

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

Inheritance diagram for FTOutlineFont:

pt3em



pt3em

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. 62) is a specialisation of the **FTFont** (p. 42) class for handling Vector Outline fonts.

pt3emSee also

FTFont (p. 42)

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.

pt3emParameters

pt3em	pt3em	pt3emfont file path.
	<i>fontFilePath</i>	

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

Open and read a font from a buffer in memory.

pt3em

Sets Error flag. The buffer is owned by the client and is NOT copied by **FTGL** (p. 23). The pointer must be valid while using **FTGL** (p. 23).

pt3emParameters

pt3em <i>pBufferBytes</i>	pt3emthe in-memory buffer
pt3em <i>buffer-SizeInBytes</i>	pt3emthe 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. 55).

pt3emParameters

pt3em <i>slot</i>	pt3emA FreeType glyph slot.
-------------------	-----------------------------

pt3emReturns

An FT***Glyph or null on failure.

Implements **FTFont** (p. 52).

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

- **FTGLOutlineFont.h**

6.13 FTOutlineGlyph Class Reference

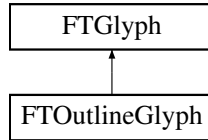
FTOutlineGlyph (p. 64) is a specialisation of **FTGlyph** (p. 55) for creating outlines.

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

Inheritance diagram for FTOutlineGlyph:

pt3em

pt3em



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.*

6.13.1 Detailed Description

FTOutlineGlyph (p. 64) is a specialisation of **FTGlyph** (p. 55) 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.

pt3emParameters

pt3emglyph	pt3emThe Freetype glyph to be processed
pt3emoutset	pt3emoutset distance
pt3emuse-DisplayList	pt3emEnable or disable the use of Display Lists for this glyph true turns ON display lists. false turns OFF display lists.

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

Destructor.

pt3em

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.

pt3emParameters

pt3em <i>pen</i>	pt3em The current pen position.
pt3em <i>renderMode</i>	pt3em Render mode to display

pt3emReturns

The advance distance for this glyph.

Implements **FTGlyph** (p. 57).

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

- **FTOutlineGlyph.h**

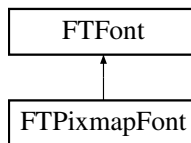
6.14 FTPixmapFont Class Reference

FTPixmapFont (p. 66) is a specialisation of the **FTFont** (p. 42) class for handling - Pixmap (Grey Scale) fonts.

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

Inheritance diagram for FTPixmapFont:

pt3em



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.

pt3em

pt3em pt3emGenerated on Sat Feb 25 2012 14:09:19 for FTGL by Doxygen

pt3em

- `~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. 66) is a specialisation of the **FTFont** (p. 42) class for handling - Pixmap (Grey Scale) fonts.

pt3emSee also

FTFont (p. 42)

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.

pt3emParameters

pt3em	pt3em	pt3emfont file path.
	<i>fontFilePath</i>	

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. 23). The pointer must be valid while using **FTGL** (p. 23).

pt3emParameters

pt3em	pt3em	pt3emthe in-memory buffer
	<i>pBufferBytes</i>	

pt3em

pt3embuffer- SizeInBytes	pt3emthe 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. 55).

pt3emParameters

pt3ept3emslot	pt3emA FreeType glyph slot.
---------------	-----------------------------

pt3emReturns

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

Implements **FTFont** (p. 52).

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

- **FTGLPixmapFont.h**

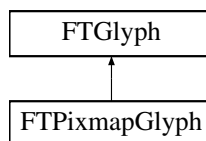
6.15 FTPixmapGlyph Class Reference

FTPixmapGlyph (p. 68) is a specialisation of **FTGlyph** (p. 55) for creating pixmaps.

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

Inheritance diagram for FTPixmapGlyph:

pt3em



pt3em

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.

6.15.1 Detailed Description

FTPixmapGlyph (p. 68) is a specialisation of **FTGlyph** (p. 55) 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

FT_GlyphSlot glyph	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

FTPoint pen	The current pen position.
int renderMode	Render mode to display

pt3em

pt3emReturns

The advance distance for this glyph.

Implements **FTGlyph** (p. 57).

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

- **FTPixmapGlyph.h**

6.16 FTPoint Class Reference

FTPoint (p. 70) 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)

pt3em

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. 70) 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.

pt3em

pt3emParameters

pt3em pt3emx	pt3emFirst component
pt3emy	pt3emSecond component
pt3emz	pt3emThird 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. 70)

pt3emParameters

pt3em pt3em <i>ft_vector</i>	pt3emA 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.

pt3emReturns

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.

pt3em

pt3emParameters

pt3em pt3em <i>multiplier</i>	pt3em
----------------------------------	-------

pt3emReturns

`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 +.

pt3emParameters

pt3em pt3em <i>point</i>	pt3em
-----------------------------	-------

pt3emReturns

`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.

pt3emParameters

pt3em pt3em <i>point</i>	pt3em
-----------------------------	-------

pt3emReturns

`this` plus `point`.

Definition at line 97 of file FTPoint.h.

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

Operator -.

pt3em

pt3emParameters

pt3em	point	pt3em
-------	-------	-------

pt3emReturns

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.

pt3emParameters

pt3em	point	pt3em
-------	-------	-------

pt3emReturns

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.

pt3emParameters

pt3em	point	pt3em	Second point
-------	-------	-------	--------------

pt3emReturns

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.

pt3em

Referenced by FTBBox::operator|=([FTPoint](#)).

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([FTPoint](#)).

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

Definition at line 250 of file FTPoint.h.

Referenced by FTBBox::operator|=([FTPoint](#)).

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([FTPoint](#)).

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

Definition at line 251 of file FTPoint.h.

Referenced by FTBBox::operator|=([FTPoint](#)).

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([FTPoint](#)).

pt3em**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.

pt3emParameters

pt3em pt3ema	pt3em
pt3emb	pt3em

pt3emReturns

true if a & b are not equal

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

Operator * Scalar multiplication.

pt3emParameters

pt3em point	pt3em
pt3em multiplier	pt3em

pt3emReturns

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.

pt3emParameters

pt3em pt3ema	pt3emFirst vector.
pt3emb	pt3emSecond vector.

pt3em

pt3emReturns

$a \cdot b$ 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.

pt3emParameters

pt3em	pt3ema	pt3em
	pt3emb	pt3em

pt3emReturns

true if a & b are equal

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

- **FTPoint.h**

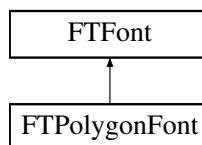
6.17 FTPolygonFont Class Reference

FTPolygonFont (p. 77) is a specialisation of the **FTFont** (p. 42) class for handling tessellated Polygon Mesh fonts.

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

Inheritance diagram for FTPolygonFont:

pt3em



Public Member Functions

- **FTPolygonFont** (const char *fontFilePath)
Open and read a font file.
- **FTPolygonFont** (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

pt3em

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. 77) is a specialisation of the **FTFont** (p. 42) class for handling tessellated Polygon Mesh fonts.

pt3emSee also

FTFont (p. 42)

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.

pt3emParameters

pt3em	pt3em	pt3emfont file path.
	<i>fontFilePath</i>	

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. 23). The pointer must be valid while using **FTGL** (p. 23).

pt3emParameters

<code>pBufferBytes</code>	the in-memory buffer
<code>buffer-SizeInBytes</code>	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. 55).

Parameters

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

Returns

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

Implements **FTFont** (p. 52).

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

- **FTGLPolygonFont.h**

6.18 FTPolygonGlyph Class Reference

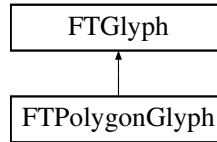
FTPolygonGlyph (p. 79) is a specialisation of **FTGlyph** (p. 55) for creating tessellated polygon glyphs.

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

Inheritance diagram for FTPolygonGlyph:

pt3em

pt3em



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.

6.18.1 Detailed Description

FTPolygonGlyph (p. 79) is a specialisation of **FTGlyph** (p. 55) 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.

pt3emParameters

pt3em <i>glyph</i>	pt3em The FreeType glyph to be processed
pt3em <i>outset</i>	pt3em The outset distance
pt3em <i>use-DisplayList</i>	pt3em 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.

pt3em

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.

pt3emParameters

pt3em <i>pen</i>	pt3em The current pen position.
pt3em <i>renderMode</i>	pt3em Render mode to display

pt3emReturns

The advance distance for this glyph.

Implements **FTGlyph** (p. 57).

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

- **FTPolyGlyph.h**

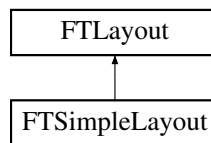
6.19 FTSimpleLayout Class Reference

FTSimpleLayout (p. 81) is a specialisation of **FTLayout** (p. 59) for simple text boxes.

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

Inheritance diagram for FTSimpleLayout:

pt3em



Public Member Functions

- **FTSimpleLayout** ()
Initializes line spacing to 1.0, alignment to ALIGN_LEFT and wrap to 100.0.
- **~FTSimpleLayout** ()
Destructor.

pt3em

- virtual **FTBBBox BBox** (const char *string, const int len=-1, **FTPoint** position=**FTPoint**())
Get the bounding box for a formatted string.
- virtual **FTBBBox 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

6.19.1 Detailed Description

FTSimpleLayout (p. 81) is a specialisation of **FTLayout** (p. 59) for simple text boxes.

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

pt3emSee also

FTLayout (p. 59)

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

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. 60).

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

Returns

The corresponding bounding box.

Implements **FTLayout** (p. 61).

pt3em**6.19.3.3 FTGL::TextAlignment FTSimpleLayout::GetAlignment () const****pt3emReturns**

The text alignment mode.

6.19.3.4 FTFont* FTSimpleLayout::GetFont ()**pt3emReturns**

The current font.

6.19.3.5 float FTSimpleLayout::GetLineLength () const**pt3emReturns**

The current line length.

6.19.3.6 float FTSimpleLayout::GetLineSpacing () const**pt3emReturns**

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.

pt3emParameters

pt3em <i>string</i>	pt3em'C' style string to be output.
pt3em <i>len</i>	pt3emThe length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
pt3em <i>position</i>	pt3emThe pen position of the first character (optional).
pt3em <i>renderMode</i>	pt3emRender mode to display (optional)

Implements **FTLayout** (p. 61).

Render

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. 62).

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.

pt3em

pt3emParameters

pt3em pt3em <i>LineLength</i>	pt3emThe new line length.
----------------------------------	---------------------------

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

Sets the line height.

pt3emParameters

pt3em pt3em <i>LineSpacing</i>	pt3emThe 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**

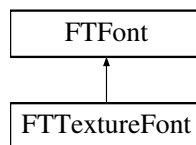
6.20 FTTextureFont Class Reference

FTTextureFont (p. 86) is a specialisation of the **FTFont** (p. 42) class for handling Texture mapped fonts.

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

Inheritance diagram for FTTextureFont:

pt3em



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.

pt3em

Protected Member Functions

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

6.20.1 Detailed Description

FTTextureFont (p. 86) is a specialisation of the **FTFont** (p. 42) class for handling Texture mapped fonts.

pt3emSee also

FTFont (p. 42)

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.

pt3emParameters

pt3em	pt3em	pt3emfont file path.
	<i>fontFilePath</i>	

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. 23). The pointer must be valid while using **FTGL** (p. 23).

pt3emParameters

pt3em	pt3em	pt3emthe in-memory buffer
	<i>pBufferBytes</i>	
pt3em	<i>buffer-</i> <i>SizeInBytes</i>	pt3emthe length of the buffer in bytes

pt3em

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.

6.21.1 Detailed Description

FTTextureGlyph (p.88) is a specialisation of **FTGlyph** (p.55) 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.

pt3emParameters

pt3emglyph	pt3emThe Freetype glyph to be processed
pt3emid	pt3emThe id of the texture that this glyph will be drawn in
pt3emxOffset	pt3emThe x offset into the parent texture to draw this glyph
pt3emyOffset	pt3emThe y offset into the parent texture to draw this glyph
pt3emwidth	pt3emThe width of the parent texture
pt3emheight	pt3emThe height (number of rows) of the parent texture

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

Destructor.

6.21.3 Member Function Documentation

pt3em

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

Render this glyph at the current pen position.

pt3emParameters

pt3em <i>pen</i>	pt3emThe current pen position.
pt3em <i>renderMode</i>	pt3emRender mode to display

pt3emReturns

The advance distance for this glyph.

Implements **FTGlyph** (p. 57).

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

- **FTTextureGlyph.h**

pt3em

pt3em

pt3em

pt3em

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. 25) is a convenience class for handling bounding boxes.

7.3 FTBitmapGlyph.h File Reference

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

Data Structures

- class **FTBitmapGlyph**

FTBitmapGlyph (p. 30) is a specialisation of *FTGlyph* (p. 55) for creating bitmaps.

Functions

- **FTGLglyph * ftgICreateBitmapGlyph** (FT_GlyphSlot glyph)

Create a specialisation of *FTGLglyph* for creating bitmaps.

pt3em

pt3em

pt3em

pt3em

7.3.1 Function Documentation

7.3.1.1 FTGLglyph* ftglCreateBitmapGlyph (FT_GlyphSlot glyph)

Create a specialisation of FTGLglyph for creating bitmaps.

pt3emParameters

pt3emglyph	pt3emThe Freetype glyph to be processed
------------	---

pt3emReturns

An FTGLglyph* object.

7.4 FTBuffer.h File Reference

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

Data Structures

- class **FTBuffer**
FTBuffer (p. 32) is a helper class for pixel buffers.

7.5 FTBufferFont.h File Reference

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

Data Structures

- class **FTBufferFont**
FTBufferFont (p. 34) is a specialisation of the *FTFont* (p. 42) class for handling memory buffer fonts.

Functions

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

pt3em

7.5.1 Function Documentation

7.5.1.1 FTGLfont* ftglCreateBufferFont (const char * file)

Create a specialised FTGLfont object for handling memory buffer fonts.

pt3emParameters

pt3em	pt3em	pt3em
file	The font file name.	

pt3emReturns

An FTGLfont* object.

pt3emSee also

FTGLfont (p. 96)

7.6 FTBufferGlyph.h File Reference

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

Data Structures

- class **FTBufferGlyph**
FTBufferGlyph (p. 37) is a specialisation of **FTGlyph** (p. 55) for memory buffer rendering.

7.7 FTEXtrdGlyph.h File Reference

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

Data Structures

- class **FTEXtrudeGlyph**
FTEXtrudeGlyph (p. 41) is a specialisation of **FTGlyph** (p. 55) for creating tessellated extruded polygon glyphs.

Defines

- #define **FTEXtrdGlyph FTEXtrudeGlyph**

pt3em

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 Define Documentation

7.7.1.1 #define FTExtrdGlyph FTExtrudeGlyph

Definition at line 77 of file FTExtrdGlyph.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.

pt3emParameters

pt3emglyph	pt3emThe FreeType glyph to be processed
pt3emdepth	pt3emThe distance along the z axis to extrude the glyph
pt3emfrontOutset	pt3emoutset contour size
pt3embackOutset	pt3emoutset contour size
pt3emuse-DisplayList	pt3emEnable or disable the use of Display Lists for this glyph <code>true</code> turns ON display lists. <code>false</code> turns OFF display lists.

pt3emReturns

An FTGLglyph* object.

7.8 FTFont.h File Reference

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

Data Structures

- class **FTFont**
FTFont (p. 42) is the public interface for the *FTGL* (p. 23) library.

pt3em

Typedefs

- typedef struct _FTGLfont **FTGLfont**

Functions

- **FTGLfont * ftglCreateCustomFont** (char const *fontFilePath, void *data, **FTGLglyph** *(*makeglyphCallback)(FT_GlyphSlot, void *))
*Create a custom **FTGL** (p. 23) font object.*
- void **ftglDestroyFont** (**FTGLfont** *font)
*Destroy an **FTGL** (p. 23) 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. 23).*
- 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)

pt3em

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.

pt3emParameters

pt3emfont	pt3emAn FTGLfont* object.
pt3emdata	pt3emThe in-memory buffer.
pt3emsize	pt3emThe length of the buffer in bytes.

pt3emReturns

- 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.

pt3emParameters

pt3emfont	pt3emAn FTGLfont* object.
pt3empath	pt3emAuxilliary font file path.

pt3em

pt3emReturns

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. 23) font object.

pt3emParameters

pt3em <i>fontFilePath</i>	pt3emThe font file name.
pt3em <i>data</i>	pt3emA pointer to private data that will be passed to callbacks.
pt3em <i>makeglyph-Callback</i>	pt3emA glyph-making callback function.

pt3emReturns

An FTGLfont* object.

7.8.2.4 void ftglDestroyFont (FTGLfont * *font*)

Destroy an **FTGL** (p. 23) font object.

pt3emParameters

pt3em <i>font</i>	pt3emAn FTGLfont* object.
-------------------	---------------------------

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

Get the advance width for a string.

pt3emParameters

pt3em <i>font</i>	pt3emAn FTGLfont* object.
pt3em <i>string</i>	pt3emA char string.

pt3em

pt3emReturns

Advance width

7.8.2.6 float ftgIGetFontAscender (FTGLfont * font)

Get the global ascender height for the face.

pt3emParameters

pt3emfont	An FTGLfont* object.
-----------	----------------------

pt3emReturns

Ascender height

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

Get the bounding box for a string.

pt3emParameters

pt3emfont	An FTGLfont* object.
pt3emstring	A char buffer
pt3emlen	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
pt3embounds	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 ftgIGetFontCharMapCount (FTGLfont * font)

Get the number of character maps in this face.

pt3emParameters

pt3emfont	An FTGLfont* object.
-----------	----------------------

pt3emReturns

character map count.

pt3em**7.8.2.9 FT_Encoding* ftglGetFontCharMapList (FTGLfont * font)**

Get a list of character maps in this face.

pt3emParameters

pt3em font	pt3em An FTGLfont* object.
------------	----------------------------

pt3emReturns

pointer to the first encoding.

7.8.2.10 float ftglGetFontDescender (FTGLfont * font)

Gets the global descender height for the face.

pt3emParameters

pt3em font	pt3em An FTGLfont* object.
------------	----------------------------

pt3emReturns

Descender height

7.8.2.11 FT_Error ftglGetFontError (FTGLfont * font)

Query a font for errors.

pt3emParameters

pt3em font	pt3em An FTGLfont* object.
------------	----------------------------

pt3emReturns

The current error code.

7.8.2.12 unsigned int ftglGetFontFaceSize (FTGLfont * font)

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

pt3em

pt3emParameters

pt3em	pt3emfont	pt3emAn FTGLfont* object.
-------	-----------	---------------------------

pt3emReturns

face size

7.8.2.13 float ftGLGetFontLineHeight (FTGLfont * font)

Gets the line spacing for the font.

pt3emParameters

pt3em	pt3emfont	pt3emAn FTGLfont* object.
-------	-----------	---------------------------

pt3emReturns

Line height

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

Render a string of characters.

pt3emParameters

pt3em	pt3emfont	pt3emAn FTGLfont* object.
	pt3emstring	pt3emChar string to be output.
	pt3emmode	pt3emRender mode to display.

7.8.2.15 int ftGLSetFontCharMap (FTGLfont * font, FT_Encoding encoding)

Set the character map for the face.

pt3emParameters

pt3em	pt3emfont	pt3emAn FTGLfont* object.
	pt3emencoding	pt3emFreetype enumerate for char map code.

pt3em

pt3emReturns

1 if charmap was valid and set correctly.

7.8.2.16 void ftgISetFontDepth (FTGLfont * font, float depth)

Set the extrusion distance for the font.

Only implemented by **FTEXtrudeFont** (p. 38).

pt3emParameters

pt3emfont	pt3emAn FTGLfont* object.
pt3emdepth	pt3emThe extrusion distance.

7.8.2.17 void ftgISetFontDisplayList (FTGLfont * font, int useList)

Enable or disable the use of Display Lists inside **FTGL** (p. 23).

pt3emParameters

pt3emfont	pt3emAn FTGLfont* object.
pt3emuseList	pt3em1 turns ON display lists. 0 turns OFF display lists.

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

Set the char size for the current face.

pt3emParameters

pt3emfont	pt3emAn FTGLfont* object.
pt3emsize	pt3emThe face size in points (1/72 inch).
pt3emres	pt3emThe resolution of the target device, or 0 to use the default value of 72.

pt3emReturns

1 if size was set correctly.

pt3em

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

Set the outset distance for the font.

Only **FTOutlineFont** (p. 62), **FTPolygonFont** (p. 77) and **FTEXtrudeFont** (p. 38) implement front outset. Only **FTEXtrudeFont** (p. 38) implements back outset.

pt3emParameters

pt3em <i>font</i>	pt3em An FTGLfont* object.
pt3em <i>front</i>	pt3em The front outset distance.
pt3em <i>back</i>	pt3em 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/FTGLExtrd-
Font.h> #include <FTGL/FTGLOutlineFont.h> #include <FTGL/-
FTGLPixmapFont.h> #include <FTGL/FTGLPolygonFont.h> #include
<FTGL/FTGLTextureFont.h> #include <FTGL/FTLayout.h> #include
<FTGL/FTSimpleLayout.h>
```

Namespaces

- namespace **FTGL**

Defines

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

pt3em

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 Define 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>
```

pt3em

Data Structures

- class **FTBitmapFont**

***FTBitmapFont** (p. 28) is a specialisation of the **FTFont** (p. 42) class for handling - Bitmap fonts.*

Defines

- #define **FTGLBitmapFont FTBitmapFont**

Functions

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

Create a specialised FTGLfont object for handling bitmap fonts.

7.11.1 Define 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.

pt3emParameters

pt3emfile	pt3emThe font file name.
-----------	--------------------------

pt3emReturns

An FTGLfont* object.

See also

See also

FTGLfont (p. 96)

7.12 FTGLExtrdFont.h File Reference

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

Data Structures

- class **FTExtrudeFont**

FTExtrudeFont (p. 38) is a specialisation of the *FTFont* (p. 42) class for handling extruded Polygon fonts.

Defines

- #define **FTGLExtrdFont FTExtrudeFont**

Functions

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

Create a specialised FTGLfont object for handling extruded polygon fonts.

7.12.1 Define Documentation

7.12.1.1 #define FTGLExtrdFont FTExtrudeFont

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 polygon fonts.

Parameters

pt3em	pt3em	pt3em
-------	-------	-------

The font file name.

pt3em
pt3emReturns

An FTGLfont* object.

pt3emSee also

FTGLfont (p. 96)

ftglCreatePolygonFont (p. 109)

7.13 FTGLOutlineFont.h File Reference

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

Data Structures

- class **FTOutlineFont**

FTOutlineFont (p. 62) is a specialisation of the *FTFont* (p. 42) class for handling - Vector Outline fonts.

Defines

- #define **FTGLOutlineFont FTOutlineFont**

Functions

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

Create a specialised FTGLfont object for handling vector outline fonts.

7.13.1 Define 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

Parameters

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

Returns

An FTGLfont* object.

See also

FTGLfont (p. 96)

7.14 FTGLPixmapFont.h File Reference

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

Data Structures

- class **FTPixmapFont**

FTPixmapFont (p. 66) is a specialisation of the *FTFont* (p. 42) class for handling - Pixmap (Grey Scale) fonts.

Defines

- #define **FTGLPixmapFont FTPixmapFont**

Functions

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

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

7.14.1 Define Documentation

7.14.1.1 #define FTGLPixmapFont FTPixmapFont

Definition at line 84 of file FTGLPixmapFont.h.

pt3em

7.14.2 Function Documentation

7.14.2.1 FTGLfont* ftglCreatePixmapFont (const char * file)

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

pt3emParameters

pt3em	pt3em	pt3em
pt3em	pt3em	pt3em

pt3emReturns

An FTGLfont* object.

pt3emSee also

FTGLfont (p. 96)

7.15 FTGLPolygonFont.h File Reference

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

Data Structures

- class **FTPolygonFont**
FTPolygonFont (p. 77) is a specialisation of the **FTFont** (p. 42) class for handling tessellated Polygon Mesh fonts.

Defines

- #define **FTGLPolygonFont FTPolygonFont**

Functions

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

7.15.1 Define Documentation

7.15.1.1 #define FTGLPolygonFont FTPolygonFont

Definition at line 84 of file FTGLPolygonFont.h.

pt3em**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

file	The font file name.
-------------	---------------------

Returns

An FTGLfont* object.

See also

FTGLfont (p. 96)

7.16 FTGLTextureFont.h File Reference

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

Data Structures

- class **FTTextureFont**
FTTextureFont (p. 86) is a specialisation of the *FTFont* (p. 42) class for handling - Texture mapped fonts.

Defines

- #define **FTGLTextureFont FTTextureFont**

Functions

- **FTGLfont * ftglCreateTextureFont** (const char *file)
Create a specialised FTGLfont object for handling texture-mapped fonts.

7.16.1 Define Documentation**7.16.1.1 #define FTGLTextureFont FTTextureFont**

Definition at line 84 of file FTGLTextureFont.h.

pt3em**7.16.2 Function Documentation****7.16.2.1 FTGLfont* ftglCreateTextureFont (const char * file)**

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

pt3emParameters

pt3empt3emfile	pt3emThe font file name.
----------------	--------------------------

pt3emReturns

An FTGLfont* object.

pt3emSee also

FTGLfont (p. 96)

7.17 FTGlyph.h File Reference

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

Data Structures

- class **FTGlyph**

FTGlyph (p. 55) is the base class for *FTGL* (p. 23) glyphs.

Typedefs

- typedef struct _FTGLglyph **FTGLglyph**

Functions

- **FTGLglyph * ftglCreateCustomGlyph (FTGLglyph *base, void *data, void(*render-Callback)(FTGLglyph *, void *, FTGL_DOUBLE, FTGL_DOUBLE, int, FTGL_DOUBLE *, FTGL_DOUBLE *), void(*destroyCallback)(FTGLglyph *, void *))**

Create a custom FTGL (p. 23) glyph object.

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

Destroy an FTGL (p. 23) glyph object.

pt3em

- 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. 23) glyph object.

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

pt3emParameters

pt3embase	pt3emThe base FTGLglyph* to subclass.
pt3emdata	pt3emA pointer to private data that will be passed to callbacks.
pt3emrender-Callback	pt3emA rendering callback function.
pt3emdestroy-Callback	pt3emA callback function to be called upon destruction.

pt3emReturns

An FTGLglyph* object.

pt3em

7.17.2.2 void ftglDestroyGlyph (FTGLglyph * *glyph*)

Destroy an **FTGL** (p. 23) glyph object.

pt3emParameters

pt3em	<i>glyph</i>	pt3emAn FTGLglyph* object.
-------	--------------	----------------------------

7.17.2.3 float ftglGetGlyphAdvance (FTGLglyph * *glyph*)

Return the advance for a glyph.

pt3emParameters

pt3em	<i>glyph</i>	pt3emAn FTGLglyph* object.
-------	--------------	----------------------------

pt3emReturns

The advance's X component.

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

Return the bounding box for a glyph.

pt3emParameters

pt3em	<i>glyph</i>	pt3emAn FTGLglyph* object.
	<i>bounds</i>	pt3emAn 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.

pt3emParameters

pt3em	<i>glyph</i>	pt3emAn FTGLglyph* object.
-------	--------------	----------------------------

pt3em

pt3emReturns

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.

pt3emParameters

pt3em <i>glyph</i>	pt3em An FTGLglyph* object.
pt3em <i>penx</i>	pt3em The current pen's X position.
pt3em <i>peny</i>	pt3em The current pen's Y position.
pt3em <i>renderMode</i>	pt3em Render mode to display
pt3em <i>advancex</i>	pt3em A pointer to an FTGL_DOUBLE where to write the advance's X component.
pt3em <i>advancey</i>	pt3em 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. 59) is the interface for layout managers that render text.

Typedefs

- typedef struct _FTGLlayout **FTGLlayout**

Functions

- void **ftglDestroyLayout** (FTGLlayout *layout)
Destroy an FTGL (p. 23) *layout object.*
- void **ftglGetLayoutBBox** (FTGLlayout *layout, const char *string, float bounds[6])

pt3em

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. 23) layout object.

pt3emParameters

pt3em	pt3em	An FTGLlayout* object.
-------	-------	------------------------

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

Get the bounding box for a string.

pt3emParameters

pt3em	pt3em	An FTGLlayout* object.
pt3em	pt3em	A char buffer
pt3em	pt3em	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.

pt3em

pt3emParameters

pt3em	pt3emAn FTGLLayout* object.
-------	-----------------------------

pt3emReturns

The current error code.

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

Render a string of characters.

pt3emParameters

pt3em	pt3emAn FTGLLayout* object.
pt3em	pt3emChar string to be output.
pt3em	pt3emRender mode to display.

7.19 FTOutlineGlyph.h File Reference

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

Data Structures

- class **FTOutlineGlyph**

FTOutlineGlyph (p. 64) is a specialisation of *FTGlyph* (p. 55) 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.

pt3em

pt3emParameters

pt3emglyph	pt3emThe Freetype glyph to be processed
pt3emoutset	pt3emoutset contour size
pt3emuse-DisplayList	pt3emEnable or disable the use of Display Lists for this glyph <code>true</code> turns ON display lists. <code>false</code> turns OFF display lists.

pt3emReturns

An FTGLglyph* object.

7.20 FTPixmapGlyph.h File Reference

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

Data Structures

- class **FTPixmapGlyph**
FTPixmapGlyph (p. 68) is a specialisation of *FTGlyph* (p. 55) for creating pixmaps.

Functions

- **FTGLglyph * ftgCreatePixmapGlyph** (FT_GlyphSlot glyph)
Create a specialisation of FTGLglyph for creating pixmaps.

7.20.1 Function Documentation

7.20.1.1 FTGLglyph* ftgCreatePixmapGlyph (FT_GlyphSlot glyph)

Create a specialisation of FTGLglyph for creating pixmaps.

pt3emParameters

pt3emglyph	pt3emThe Freetype glyph to be processed
------------	---

pt3em
pt3emReturns

An FTGLglyph* object.

7.21 FTPoint.h File Reference

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

Data Structures

- class **FTPoint**

FTPoint (p. 70) 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. 79) is a specialisation of *FTGlyph* (p. 55) for creating tessellated polygon glyphs.

Defines

- #define **FTPolyGlyph FTPolygonGlyph**

Functions

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

Create a specialisation of FTGLglyph for creating tessellated polygon glyphs.

7.22.1 Define Documentation

7.22.1.1 #define FTPolyGlyph FTPolygonGlyph

Definition at line 74 of file FTPolyGlyph.h.

pt3em

7.22.2 Function Documentation

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

Create a specialisation of FTGLglyph for creating tessellated polygon glyphs.

pt3emParameters

pt3emglyph	pt3emThe Freetype glyph to be processed
pt3emoutset	pt3emoutset contour size
pt3emuse- DisplayList	pt3emEnable or disable the use of Display Lists for this glyph <code>true</code> turns ON display lists. <code>false</code> turns OFF display lists.

pt3emReturns

An FTGLglyph* object.

7.23 FTSimpleLayout.h File Reference

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

Data Structures

- class **FTSimpleLayout**

FTSimpleLayout (p. 81) is a specialisation of *FTLayout* (p. 59) for simple text boxes.

Functions

- **FTGLlayout * ftgICreateSimpleLayout** (void)
- void **ftgISetLayoutFont** (FTGLlayout *, FTGLfont *)
- **FTGLfont * ftgIGetLayoutFont** (FTGLlayout *)
- void **ftgISetLayoutLineLength** (FTGLlayout *, const float)
- float **ftgIGetLayoutLineLength** (FTGLlayout *)
- void **ftgISetLayoutAlignment** (FTGLlayout *, const int)
- int **ftgIGetLayoutAligment** (FTGLlayout *)
- void **ftgISetLayoutLineSpacing** (FTGLlayout *, const float)
- float **ftgIGetLayoutLineSpacing** (FTGLlayout *)

pt3em

7.23.1 Function Documentation

7.23.1.1 FTGLLayout* ftglCreateSimpleLayout (void)

7.23.1.2 int ftglGetLayoutAligment (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 ftglSetLayoutAligment (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. 88) is a specialisation of *FTGlyph* (p. 55) 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.

pt3em

pt3emParameters

pt3em <i>glyph</i>	pt3emThe FreeType glyph to be processed.
pt3em <i>id</i>	pt3emThe id of the texture that this glyph will be drawn in.
pt3em <i>xOffset</i>	pt3emThe x offset into the parent texture to draw this glyph.
pt3em <i>yOffset</i>	pt3emThe y offset into the parent texture to draw this glyph.
pt3em <i>width</i>	pt3emThe width of the parent texture.
pt3em <i>height</i>	pt3emThe height (number of rows) of the parent texture.

pt3emReturns

An FTGLglyph* object.

7.25 projects_using_ftgl.txt File Reference

7.26 tutorial.dox File Reference