SFML

* [Main Page](http://docs.google.com/index.htm)
* [Modules](http://docs.google.com/modules.htm)
* [Classes](http://docs.google.com/annotated.htm)
* [Files](http://docs.google.com/files.htm)
* [File List](http://docs.google.com/files.htm)
* [include](http://docs.google.com/dir_f3190241575fd2bd132a392ae6942f4a.htm)
* [SFML](http://docs.google.com/dir_692f376662c82a26cfe4cfa3aceebe24.htm)
* [Graphics](http://docs.google.com/dir_aaa96c3797a59111c2945d0d638ce5cf.htm)

Font.hpp

1

2 //

3 // SFML - Simple and Fast Multimedia Library

4 // Copyright (C) 2007-2013 Laurent Gomila (laurent.gom@gmail.com)

5 //

6 // This software is provided 'as-is', without any express or implied warranty.

7 // In no event will the authors be held liable for any damages arising from the use of this software.

8 //

9 // Permission is granted to anyone to use this software for any purpose,

10 // including commercial applications, and to alter it and redistribute it freely,

11 // subject to the following restrictions:

12 //

13 // 1. The origin of this software must not be misrepresented;

14 // you must not claim that you wrote the original software.

15 // If you use this software in a product, an acknowledgment

16 // in the product documentation would be appreciated but is not required.

17 //

18 // 2. Altered source versions must be plainly marked as such,

19 // and must not be misrepresented as being the original software.

20 //

21 // 3. This notice may not be removed or altered from any source distribution.

22 //

24

25 #ifndef SFML\_FONT\_HPP

26 #define SFML\_FONT\_HPP

27

29 // Headers

31 #include <SFML/Graphics/Export.hpp>

32 #include <SFML/Graphics/Glyph.hpp>

33 #include <SFML/Graphics/Texture.hpp>

34 #include <SFML/Graphics/Rect.hpp>

35 #include <SFML/System/Vector2.hpp>

36 #include <SFML/System/String.hpp>

37 #include <map>

38 #include <string>

39 #include <vector>

40

41

42 namespace sf

43 {

44 class InputStream;

45

[50](http://docs.google.com/classsf_1_1Font.htm) class SFML\_GRAPHICS\_API [Font](http://docs.google.com/classsf_1_1Font.htm)

51 {

52 public :

53

60  [Font](http://docs.google.com/classsf_1_1Font.htm)();

61

68  [Font](http://docs.google.com/classsf_1_1Font.htm)(const [Font](http://docs.google.com/classsf_1_1Font.htm)& copy);

69

76  ~[Font](http://docs.google.com/classsf_1_1Font.htm)();

77

94  bool loadFromFile(const std::string& filename);

95

113  bool loadFromMemory(const void\* data, std::size\_t sizeInBytes);

114

131  bool loadFromStream([InputStream](http://docs.google.com/classsf_1_1InputStream.htm)& stream);

132

143  const [Glyph](http://docs.google.com/classsf_1_1Glyph.htm)& getGlyph(Uint32 codePoint, unsigned int characterSize, bool bold) const;

144

161  int getKerning(Uint32 first, Uint32 second, unsigned int characterSize) const;

162

174  int getLineSpacing(unsigned int characterSize) const;

175

188  const [Texture](http://docs.google.com/classsf_1_1Texture.htm)& getTexture(unsigned int characterSize) const;

189

198  [Font](http://docs.google.com/classsf_1_1Font.htm)& operator =(const [Font](http://docs.google.com/classsf_1_1Font.htm)& right);

199

200 private :

201

206  struct Row

207  {

208  Row(unsigned int rowTop, unsigned int rowHeight) : width(0), top(rowTop), height(rowHeight) {}

209

210  unsigned int width;

211  unsigned int top;

212  unsigned int height;

213  };

214

216  // Types

218  typedef std::map<Uint32, Glyph> GlyphTable;

219

224  struct Page

225  {

226  Page();

227

228  GlyphTable glyphs;

229  [sf::Texture](http://docs.google.com/classsf_1_1Texture.htm) texture;

230  unsigned int nextRow;

231  std::vector<Row> rows;

232  };

233

238  void cleanup();

239

250  [Glyph](http://docs.google.com/classsf_1_1Glyph.htm) loadGlyph(Uint32 codePoint, unsigned int characterSize, bool bold) const;

251

262  [IntRect](http://docs.google.com/classsf_1_1Rect.htm) findGlyphRect(Page& page, unsigned int width, unsigned int height) const;

263

272  bool setCurrentSize(unsigned int characterSize) const;

273

275  // Types

277  typedef std::map<unsigned int, Page> PageTable;

278

280  // Member data

282  void\* m\_library;

283  void\* m\_face;

284  void\* m\_streamRec;

285  int\* m\_refCount;

286  mutable PageTable m\_pages;

287  mutable std::vector<Uint8> m\_pixelBuffer;

288 };

289

290 } // namespace sf

291

292

293 #endif // SFML\_FONT\_HPP

294

295

Copyright � Laurent Gomila  ::  Documentation generated by [doxygen](http://www.doxygen.org/)  ::