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)

Texture.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\_TEXTURE\_HPP

26 #define SFML\_TEXTURE\_HPP

27

29 // Headers

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

32 #include <SFML/Graphics/Image.hpp>

33 #include <SFML/Window/GlResource.hpp>

34

35

36 namespace sf

37 {

38 class Window;

39 class RenderTarget;

40 class RenderTexture;

41 class InputStream;

42

[47](http://docs.google.com/classsf_1_1Texture.htm) class SFML\_GRAPHICS\_API [Texture](http://docs.google.com/classsf_1_1Texture.htm) : [GlResource](http://docs.google.com/classsf_1_1GlResource.htm)

48 {

49 public :

50

[55](http://docs.google.com/classsf_1_1Texture.htm#aa6fd3bbe3c334b3c4428edfb2765a82e)  enum [CoordinateType](http://docs.google.com/classsf_1_1Texture.htm#aa6fd3bbe3c334b3c4428edfb2765a82e)

56  {

[57](http://docs.google.com/classsf_1_1Texture.htm#aa6fd3bbe3c334b3c4428edfb2765a82ea69d6228950882e4d68be4ba4dbe7df73)  [Normalized](http://docs.google.com/classsf_1_1Texture.htm#aa6fd3bbe3c334b3c4428edfb2765a82ea69d6228950882e4d68be4ba4dbe7df73),

[58](http://docs.google.com/classsf_1_1Texture.htm#aa6fd3bbe3c334b3c4428edfb2765a82ea6372f9c3a10203a7a69d8d5da59d82ff)  Pixels

59  };

60

61 public :

62

69  [Texture](http://docs.google.com/classsf_1_1Texture.htm)();

70

77  [Texture](http://docs.google.com/classsf_1_1Texture.htm)(const [Texture](http://docs.google.com/classsf_1_1Texture.htm)& copy);

78

83  ~[Texture](http://docs.google.com/classsf_1_1Texture.htm)();

84

96  bool create(unsigned int width, unsigned int height);

97

127  bool loadFromFile(const std::string& filename, const [IntRect](http://docs.google.com/classsf_1_1Rect.htm)& area = [IntRect](http://docs.google.com/classsf_1_1Rect.htm)());

128

159  bool loadFromMemory(const void\* data, std::size\_t size, const [IntRect](http://docs.google.com/classsf_1_1Rect.htm)& area = [IntRect](http://docs.google.com/classsf_1_1Rect.htm)());

160

190  bool loadFromStream([sf::InputStream](http://docs.google.com/classsf_1_1InputStream.htm)& stream, const [IntRect](http://docs.google.com/classsf_1_1Rect.htm)& area = [IntRect](http://docs.google.com/classsf_1_1Rect.htm)());

191

214  bool loadFromImage(const [Image](http://docs.google.com/classsf_1_1Image.htm)& image, const [IntRect](http://docs.google.com/classsf_1_1Rect.htm)& area = [IntRect](http://docs.google.com/classsf_1_1Rect.htm)());

215

222  [Vector2u](http://docs.google.com/classsf_1_1Vector2.htm) getSize() const;

223

237  [Image](http://docs.google.com/classsf_1_1Image.htm) copyToImage() const;

238

255  void update(const Uint8\* pixels);

256

277  void update(const Uint8\* pixels, unsigned int width, unsigned int height, unsigned int x, unsigned int y);

278

297  void update(const [Image](http://docs.google.com/classsf_1_1Image.htm)& image);

298

314  void update(const [Image](http://docs.google.com/classsf_1_1Image.htm)& image, unsigned int x, unsigned int y);

315

334  void update(const [Window](http://docs.google.com/classsf_1_1Window.htm)& window);

335

351  void update(const [Window](http://docs.google.com/classsf_1_1Window.htm)& window, unsigned int x, unsigned int y);

352

367  void setSmooth(bool smooth);

368

377  bool isSmooth() const;

378

401  void setRepeated(bool repeated);

402

411  bool isRepeated() const;

412

421  [Texture](http://docs.google.com/classsf_1_1Texture.htm)& operator =(const [Texture](http://docs.google.com/classsf_1_1Texture.htm)& right);

422

454  static void bind(const [Texture](http://docs.google.com/classsf_1_1Texture.htm)\* texture, CoordinateType coordinateType = Normalized);

455

466  static unsigned int getMaximumSize();

467

468 private :

469

470  friend class [RenderTexture](http://docs.google.com/classsf_1_1RenderTexture.htm);

471  friend class [RenderTarget](http://docs.google.com/classsf_1_1RenderTarget.htm);

472

486  static unsigned int getValidSize(unsigned int size);

487

489  // Member data

491  [Vector2u](http://docs.google.com/classsf_1_1Vector2.htm) m\_size;

492  [Vector2u](http://docs.google.com/classsf_1_1Vector2.htm) m\_actualSize;

493  unsigned int m\_texture;

494  bool m\_isSmooth;

495  bool m\_isRepeated;

496  mutable bool m\_pixelsFlipped;

497  Uint64 m\_cacheId;

498 };

499

500 } // namespace sf

501

502

503 #endif // SFML\_TEXTURE\_HPP

504

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