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)
* [Class List](http://docs.google.com/annotated.htm)
* [Class Index](http://docs.google.com/classes.htm)
* [Class Hierarchy](http://docs.google.com/hierarchy.htm)
* [Class Members](http://docs.google.com/functions.htm)
* **sf**
* [Image](http://docs.google.com/classsf_1_1Image.htm)

[Public Member Functions](#_gjdgxs) | [List of all members](http://docs.google.com/classsf_1_1Image-members.htm)

sf::Image Class Reference

[Graphics module](http://docs.google.com/group__graphics.htm)

Class for loading, manipulating and saving images. [More...](http://docs.google.com/classsf_1_1Image.htm#details)

#include <[Image.hpp](http://docs.google.com/Image_8hpp_source.htm)>

| Public Member Functions | |
| --- | --- |
|  | [Image](http://docs.google.com/classsf_1_1Image.htm#abb4caf3cb167b613345ebe36fc883f12) () |
|  | Default constructor. |
|  | |
| void | [create](http://docs.google.com/classsf_1_1Image.htm#a2a67930e2fd9ad97cf004e918cf5832b) (unsigned int width, unsigned int height, const [Color](http://docs.google.com/classsf_1_1Color.htm) &color=[Color](http://docs.google.com/classsf_1_1Color.htm)(0, 0, 0)) |
|  | Create the image and fill it with a unique color. |
|  | |
| void | [create](http://docs.google.com/classsf_1_1Image.htm#a1c2b960ea12bdbb29e80934ce5268ebf) (unsigned int width, unsigned int height, const Uint8 \*pixels) |
|  | Create the image from an array of pixels. |
|  | |
| bool | [loadFromFile](http://docs.google.com/classsf_1_1Image.htm#a9e4f2aa8e36d0cabde5ed5a4ef80290b) (const std::string &filename) |
|  | Load the image from a file on disk. |
|  | |
| bool | [loadFromMemory](http://docs.google.com/classsf_1_1Image.htm#aaa6c7afa5851a51cec6ab438faa7354c) (const void \*data, std::size\_t size) |
|  | Load the image from a file in memory. |
|  | |
| bool | [loadFromStream](http://docs.google.com/classsf_1_1Image.htm#a21122ded0e8368bb06ed3b9acfbfb501) ([InputStream](http://docs.google.com/classsf_1_1InputStream.htm) &stream) |
|  | Load the image from a custom stream. |
|  | |
| bool | [saveToFile](http://docs.google.com/classsf_1_1Image.htm#aec0ed16b67df7b512aaa5c53388ba14e) (const std::string &filename) const |
|  | Save the image to a file on disk. |
|  | |
| [Vector2u](http://docs.google.com/classsf_1_1Vector2.htm) | [getSize](http://docs.google.com/classsf_1_1Image.htm#a5c3e9bebdc001c3ebf85ca97039fc86b) () const |
|  | Return the size (width and height) of the image. |
|  | |
| void | [createMaskFromColor](http://docs.google.com/classsf_1_1Image.htm#a22f13f8c242a6b38eb73cc176b37ae34) (const [Color](http://docs.google.com/classsf_1_1Color.htm) &color, Uint8 alpha=0) |
|  | Create a transparency mask from a specified color-key. |
|  | |
| void | [copy](http://docs.google.com/classsf_1_1Image.htm#ab2fa337c956f85f93377dcb52153a45a) (const [Image](http://docs.google.com/classsf_1_1Image.htm) &source, unsigned int destX, unsigned int destY, const [IntRect](http://docs.google.com/classsf_1_1Rect.htm) &sourceRect=[IntRect](http://docs.google.com/classsf_1_1Rect.htm)(0, 0, 0, 0), bool applyAlpha=false) |
|  | Copy pixels from another image onto this one. |
|  | |
| void | [setPixel](http://docs.google.com/classsf_1_1Image.htm#a9fd329b8cd7d4439e07fb5d3bb2d9744) (unsigned int x, unsigned int y, const [Color](http://docs.google.com/classsf_1_1Color.htm) &color) |
|  | Change the color of a pixel. |
|  | |
| [Color](http://docs.google.com/classsf_1_1Color.htm) | [getPixel](http://docs.google.com/classsf_1_1Image.htm#a8c8460e311dcb00557cb00a81c29163d) (unsigned int x, unsigned int y) const |
|  | Get the color of a pixel. |
|  | |
| const Uint8 \* | [getPixelsPtr](http://docs.google.com/classsf_1_1Image.htm#ac6137a608a9efaae2735c13ff259c214) () const |
|  | Get a read-only pointer to the array of pixels. |
|  | |
| void | [flipHorizontally](http://docs.google.com/classsf_1_1Image.htm#a57168e7bc29190e08bbd6c9c19f4bb2c) () |
|  | Flip the image horizontally (left <-> right) |
|  | |
| void | [flipVertically](http://docs.google.com/classsf_1_1Image.htm#a78a702a7e49d1de2dec9894da99d279c) () |
|  | Flip the image vertically (top <-> bottom) |
|  | |

## Detailed Description

Class for loading, manipulating and saving images.

[sf::Image](http://docs.google.com/classsf_1_1Image.htm) is an abstraction to manipulate images as bidimensional arrays of pixels.

The class provides functions to load, read, write and save pixels, as well as many other useful functions.

[sf::Image](http://docs.google.com/classsf_1_1Image.htm) can handle a unique internal representation of pixels, which is RGBA 32 bits. This means that a pixel must be composed of 8 bits red, green, blue and alpha channels – just like a [sf::Color](http://docs.google.com/classsf_1_1Color.htm). All the functions that return an array of pixels follow this rule, and all parameters that you pass to [sf::Image](http://docs.google.com/classsf_1_1Image.htm) functions (such as loadFromPixels) must use this representation as well.

A [sf::Image](http://docs.google.com/classsf_1_1Image.htm) can be copied, but it is a heavy resource and if possible you should always use [const] references to pass or return them to avoid useless copies.

Usage example:

// Load an image file from a file

[sf::Image](http://docs.google.com/classsf_1_1Image.htm) background;

if (!background.[loadFromFile](http://docs.google.com/classsf_1_1Image.htm#a9e4f2aa8e36d0cabde5ed5a4ef80290b)("background.jpg"))

return -1;

// Create a 20x20 image filled with black color

[sf::Image](http://docs.google.com/classsf_1_1Image.htm) image;

if (!image.[create](http://docs.google.com/classsf_1_1Image.htm#a2a67930e2fd9ad97cf004e918cf5832b)(20, 20, [sf::Color::Black](http://docs.google.com/classsf_1_1Color.htm#a77c688197b981338f0b19dc58bd2facd)))

return -1;

// Copy image1 on image2 at position (10, 10)

image.[copy](http://docs.google.com/classsf_1_1Image.htm#ab2fa337c956f85f93377dcb52153a45a)(background, 10, 10);

// Make the top-left pixel transparent

[sf::Color](http://docs.google.com/classsf_1_1Color.htm) color = image.[getPixel](http://docs.google.com/classsf_1_1Image.htm#a8c8460e311dcb00557cb00a81c29163d)(0, 0);

color.[a](http://docs.google.com/classsf_1_1Color.htm#a56dbdb47d5f040d9b78ac6a0b8b3a831) = 0;

image.[setPixel](http://docs.google.com/classsf_1_1Image.htm#a9fd329b8cd7d4439e07fb5d3bb2d9744)(0, 0, color);

// Save the image to a file

if (!image.[saveToFile](http://docs.google.com/classsf_1_1Image.htm#aec0ed16b67df7b512aaa5c53388ba14e)("result.png"))

return -1;

See Also[sf::Texture](http://docs.google.com/classsf_1_1Texture.htm)

Definition at line [46](http://docs.google.com/Image_8hpp_source.htm#l00046) of file [Image.hpp](http://docs.google.com/Image_8hpp_source.htm).

## Constructor & Destructor Documentation

| sf::Image::Image | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Default constructor.

Creates an empty image.

## Member Function Documentation

| void sf::Image::copy | ( | const [Image](http://docs.google.com/classsf_1_1Image.htm) & | *source*, |
| --- | --- | --- | --- |
|  |  | unsigned int | *destX*, |
|  |  | unsigned int | *destY*, |
|  |  | const [IntRect](http://docs.google.com/classsf_1_1Rect.htm) & | *sourceRect* = [IntRect](http://docs.google.com/classsf_1_1Rect.htm)(0, 0, 0, 0), |
|  |  | bool | *applyAlpha* = false |
|  | ) |  |  |

Copy pixels from another image onto this one.

This function does a slow pixel copy and should not be used intensively. It can be used to prepare a complex static image from several others, but if you need this kind of feature in real-time you'd better use [sf::RenderTexture](http://docs.google.com/classsf_1_1RenderTexture.htm).

If *sourceRect* is empty, the whole image is copied. If *applyAlpha* is set to true, the transparency of source pixels is applied. If it is false, the pixels are copied unchanged with their alpha value.

Parameters

| source | Source image to copy |
| --- | --- |
| destX | X coordinate of the destination position |
| destY | Y coordinate of the destination position |
| sourceRect | Sub-rectangle of the source image to copy |
| applyAlpha | Should the copy take in account the source transparency? |

| void sf::Image::create | ( | unsigned int | *width*, |
| --- | --- | --- | --- |
|  |  | unsigned int | *height*, |
|  |  | const [Color](http://docs.google.com/classsf_1_1Color.htm) & | *color* = [Color](http://docs.google.com/classsf_1_1Color.htm)(0, 0, 0) |
|  | ) |  |  |

Create the image and fill it with a unique color.

Parameters

| width | Width of the image |
| --- | --- |
| height | Height of the image |
| color | Fill color |

| void sf::Image::create | ( | unsigned int | *width*, |
| --- | --- | --- | --- |
|  |  | unsigned int | *height*, |
|  |  | const Uint8 \* | *pixels* |
|  | ) |  |  |

Create the image from an array of pixels.

The *pixel* array is assumed to contain 32-bits RGBA pixels, and have the given *width* and *height*. If not, this is an undefined behaviour. If *pixels* is null, an empty image is created.

Parameters

| width | Width of the image |
| --- | --- |
| height | Height of the image |
| pixels | Array of pixels to copy to the image |

| void sf::Image::createMaskFromColor | ( | const [Color](http://docs.google.com/classsf_1_1Color.htm) & | *color*, |
| --- | --- | --- | --- |
|  |  | Uint8 | *alpha* = 0 |
|  | ) |  |  |

Create a transparency mask from a specified color-key.

This function sets the alpha value of every pixel matching the given color to *alpha* (0 by default), so that they become transparent.

Parameters

| color | [Color](http://docs.google.com/classsf_1_1Color.htm) to make transparent |
| --- | --- |
| alpha | Alpha value to assign to transparent pixels |

| void sf::Image::flipHorizontally | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Flip the image horizontally (left <-> right)

| void sf::Image::flipVertically | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Flip the image vertically (top <-> bottom)

| [Color](http://docs.google.com/classsf_1_1Color.htm) sf::Image::getPixel | ( | unsigned int | *x*, |
| --- | --- | --- | --- |
|  |  | unsigned int | *y* |
|  | ) |  | const |

Get the color of a pixel.

This function doesn't check the validity of the pixel coordinates, using out-of-range values will result in an undefined behaviour.

Parameters

| x | X coordinate of pixel to get |
| --- | --- |
| y | Y coordinate of pixel to get |

Returns[Color](http://docs.google.com/classsf_1_1Color.htm) of the pixel at coordinates (x, y) See Also[setPixel](http://docs.google.com/classsf_1_1Image.htm#a9fd329b8cd7d4439e07fb5d3bb2d9744)

| const Uint8\* sf::Image::getPixelsPtr | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get a read-only pointer to the array of pixels.

The returned value points to an array of RGBA pixels made of 8 bits integers components. The size of the array is width \* height \* 4 ([getSize()](http://docs.google.com/classsf_1_1Image.htm#a5c3e9bebdc001c3ebf85ca97039fc86b).x \* [getSize()](http://docs.google.com/classsf_1_1Image.htm#a5c3e9bebdc001c3ebf85ca97039fc86b).y \* 4). Warning: the returned pointer may become invalid if you modify the image, so you should never store it for too long. If the image is empty, a null pointer is returned.

ReturnsRead-only pointer to the array of pixels

| [Vector2u](http://docs.google.com/classsf_1_1Vector2.htm) sf::Image::getSize | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Return the size (width and height) of the image.

ReturnsSize of the image, in pixels

| bool sf::Image::loadFromFile | ( | const std::string & | *filename* | ) |  |
| --- | --- | --- | --- | --- | --- |

Load the image from a file on disk.

The supported image formats are bmp, png, tga, jpg, gif, psd, hdr and pic. Some format options are not supported, like progressive jpeg. If this function fails, the image is left unchanged.

Parameters

| filename | Path of the image file to load |
| --- | --- |

ReturnsTrue if loading was successful See Also[loadFromMemory](http://docs.google.com/classsf_1_1Image.htm#aaa6c7afa5851a51cec6ab438faa7354c), [loadFromStream](http://docs.google.com/classsf_1_1Image.htm#a21122ded0e8368bb06ed3b9acfbfb501), [saveToFile](http://docs.google.com/classsf_1_1Image.htm#aec0ed16b67df7b512aaa5c53388ba14e)

| bool sf::Image::loadFromMemory | ( | const void \* | *data*, |
| --- | --- | --- | --- |
|  |  | std::size\_t | *size* |
|  | ) |  |  |

Load the image from a file in memory.

The supported image formats are bmp, png, tga, jpg, gif, psd, hdr and pic. Some format options are not supported, like progressive jpeg. If this function fails, the image is left unchanged.

Parameters

| data | Pointer to the file data in memory |
| --- | --- |
| size | Size of the data to load, in bytes |

ReturnsTrue if loading was successful See Also[loadFromFile](http://docs.google.com/classsf_1_1Image.htm#a9e4f2aa8e36d0cabde5ed5a4ef80290b), [loadFromStream](http://docs.google.com/classsf_1_1Image.htm#a21122ded0e8368bb06ed3b9acfbfb501)

| bool sf::Image::loadFromStream | ( | [InputStream](http://docs.google.com/classsf_1_1InputStream.htm) & | *stream* | ) |  |
| --- | --- | --- | --- | --- | --- |

Load the image from a custom stream.

The supported image formats are bmp, png, tga, jpg, gif, psd, hdr and pic. Some format options are not supported, like progressive jpeg. If this function fails, the image is left unchanged.

Parameters

| stream | Source stream to read from |
| --- | --- |

ReturnsTrue if loading was successful See Also[loadFromFile](http://docs.google.com/classsf_1_1Image.htm#a9e4f2aa8e36d0cabde5ed5a4ef80290b), [loadFromMemory](http://docs.google.com/classsf_1_1Image.htm#aaa6c7afa5851a51cec6ab438faa7354c)

| bool sf::Image::saveToFile | ( | const std::string & | *filename* | ) | const |
| --- | --- | --- | --- | --- | --- |

Save the image to a file on disk.

The format of the image is automatically deduced from the extension. The supported image formats are bmp, png, tga and jpg. The destination file is overwritten if it already exists. This function fails if the image is empty.

Parameters

| filename | Path of the file to save |
| --- | --- |

ReturnsTrue if saving was successful See Also[create](http://docs.google.com/classsf_1_1Image.htm#a2a67930e2fd9ad97cf004e918cf5832b), [loadFromFile](http://docs.google.com/classsf_1_1Image.htm#a9e4f2aa8e36d0cabde5ed5a4ef80290b), [loadFromMemory](http://docs.google.com/classsf_1_1Image.htm#aaa6c7afa5851a51cec6ab438faa7354c)

| void sf::Image::setPixel | ( | unsigned int | *x*, |
| --- | --- | --- | --- |
|  |  | unsigned int | *y*, |
|  |  | const [Color](http://docs.google.com/classsf_1_1Color.htm) & | *color* |
|  | ) |  |  |

Change the color of a pixel.

This function doesn't check the validity of the pixel coordinates, using out-of-range values will result in an undefined behaviour.

Parameters

| x | X coordinate of pixel to change |
| --- | --- |
| y | Y coordinate of pixel to change |
| color | New color of the pixel |

See Also[getPixel](http://docs.google.com/classsf_1_1Image.htm#a8c8460e311dcb00557cb00a81c29163d)

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

* [Image.hpp](http://docs.google.com/Image_8hpp_source.htm)

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