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**
* [Window](http://docs.google.com/classsf_1_1Window.htm)

[Public Member Functions](#_gjdgxs) | [Protected Member Functions](#_30j0zll) | [Static Private Member Functions](#_1fob9te) | [List of all members](http://docs.google.com/classsf_1_1Window-members.htm)

sf::Window Class Reference

[Window module](http://docs.google.com/group__window.htm)

[Window](http://docs.google.com/classsf_1_1Window.htm) that serves as a target for OpenGL rendering. [More...](http://docs.google.com/classsf_1_1Window.htm#details)

#include <[Window.hpp](http://docs.google.com/Window_2Window_8hpp_source.htm)>

Inheritance diagram for sf::Window:



| Public Member Functions | |
| --- | --- |
|  | [Window](http://docs.google.com/classsf_1_1Window.htm#a5359122166b4dc492c3d25caf08ccfc4) () |
|  | Default constructor. |
|  | |
|  | [Window](http://docs.google.com/classsf_1_1Window.htm#a1bee771baecbae6d357871929dc042a2) ([VideoMode](http://docs.google.com/classsf_1_1VideoMode.htm) mode, const [String](http://docs.google.com/classsf_1_1String.htm) &title, Uint32 style=Style::Default, const [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm) &settings=[ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm)()) |
|  | Construct a new window. |
|  | |
|  | [Window](http://docs.google.com/classsf_1_1Window.htm#a6d60912633bff9d33cf3ade4e0201de4) (WindowHandle handle, const [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm) &settings=[ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm)()) |
|  | Construct the window from an existing control. |
|  | |
| virtual | [~Window](http://docs.google.com/classsf_1_1Window.htm#ac30eb6ea5f5594204944d09d4bd69a97) () |
|  | Destructor. |
|  | |
| void | [create](http://docs.google.com/classsf_1_1Window.htm#a30e6edf2162f8dbff61023b9de5d961d) ([VideoMode](http://docs.google.com/classsf_1_1VideoMode.htm) mode, const [String](http://docs.google.com/classsf_1_1String.htm) &title, Uint32 style=Style::Default, const [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm) &settings=[ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm)()) |
|  | Create (or recreate) the window. |
|  | |
| void | [create](http://docs.google.com/classsf_1_1Window.htm#acf67483dc21f08d65c8835b3889b41b2) (WindowHandle handle, const [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm) &settings=[ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm)()) |
|  | Create (or recreate) the window from an existing control. |
|  | |
| void | [close](http://docs.google.com/classsf_1_1Window.htm#a99d1e030387b0c26f5995670504fe7b5) () |
|  | Close the window and destroy all the attached resources. |
|  | |
| bool | [isOpen](http://docs.google.com/classsf_1_1Window.htm#a5aa9c2b2b0e51d3423c2b66c80253337) () const |
|  | Tell whether or not the window is open. |
|  | |
| const [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm) & | [getSettings](http://docs.google.com/classsf_1_1Window.htm#a5a9d5c15facf25ad4d9b2b30caa0a2db) () const |
|  | Get the settings of the OpenGL context of the window. |
|  | |
| bool | [pollEvent](http://docs.google.com/classsf_1_1Window.htm#a338e996585faf82e93069858e3b531b7) ([Event](http://docs.google.com/classsf_1_1Event.htm) &event) |
|  | Pop the event on top of the event queue, if any, and return it. |
|  | |
| bool | [waitEvent](http://docs.google.com/classsf_1_1Window.htm#aaf02ab64fbc1d374eef3696df54137bc) ([Event](http://docs.google.com/classsf_1_1Event.htm) &event) |
|  | Wait for an event and return it. |
|  | |
| [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) | [getPosition](http://docs.google.com/classsf_1_1Window.htm#a2e6bc12612ea289afea8268fe37c8678) () const |
|  | Get the position of the window. |
|  | |
| void | [setPosition](http://docs.google.com/classsf_1_1Window.htm#a6c4078bfbf61c29bfc4b4732ce764f17) (const [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) &position) |
|  | Change the position of the window on screen. |
|  | |
| [Vector2u](http://docs.google.com/classsf_1_1Vector2.htm) | [getSize](http://docs.google.com/classsf_1_1Window.htm#ad2b55a731ba1680fe67292991ef1610e) () const |
|  | Get the size of the rendering region of the window. |
|  | |
| void | [setSize](http://docs.google.com/classsf_1_1Window.htm#ad6513418bb2963347cd1819a1810524d) (const [Vector2u](http://docs.google.com/classsf_1_1Vector2.htm) size) |
|  | Change the size of the rendering region of the window. |
|  | |
| void | [setTitle](http://docs.google.com/classsf_1_1Window.htm#a3b3f3513bb6be90f5cd456c20b5fd5fa) (const [String](http://docs.google.com/classsf_1_1String.htm) &title) |
|  | Change the title of the window. |
|  | |
| void | [setIcon](http://docs.google.com/classsf_1_1Window.htm#a63af61e026fba08e3153fd013620bcc0) (unsigned int width, unsigned int height, const Uint8 \*pixels) |
|  | Change the window's icon. |
|  | |
| void | [setVisible](http://docs.google.com/classsf_1_1Window.htm#a160f7f11a207603d7e99ce606e749703) (bool visible) |
|  | Show or hide the window. |
|  | |
| void | [setVerticalSyncEnabled](http://docs.google.com/classsf_1_1Window.htm#a59041c4556e0351048f8aff366034f61) (bool enabled) |
|  | Enable or disable vertical synchronization. |
|  | |
| void | [setMouseCursorVisible](http://docs.google.com/classsf_1_1Window.htm#aad3991c25e0a83afbb4d62febf9b7b14) (bool visible) |
|  | Show or hide the mouse cursor. |
|  | |
| void | [setKeyRepeatEnabled](http://docs.google.com/classsf_1_1Window.htm#aef9f2b14c10ecba8a8df95dd51c5bb73) (bool enabled) |
|  | Enable or disable automatic key-repeat. |
|  | |
| void | [setFramerateLimit](http://docs.google.com/classsf_1_1Window.htm#af4322d315baf93405bf0d5087ad5e784) (unsigned int limit) |
|  | Limit the framerate to a maximum fixed frequency. |
|  | |
| void | [setJoystickThreshold](http://docs.google.com/classsf_1_1Window.htm#aa45b8f54e29a6f59f1fc7ee66b2fab68) (float threshold) |
|  | Change the joystick threshold. |
|  | |
| bool | [setActive](http://docs.google.com/classsf_1_1Window.htm#a17ccf8ece0ce0bf2f1e6698bcfa29731) (bool active=true) const |
|  | Activate or deactivate the window as the current target for OpenGL rendering. |
|  | |
| void | [display](http://docs.google.com/classsf_1_1Window.htm#adabf839cb103ac96cfc82f781638772a) () |
|  | Display on screen what has been rendered to the window so far. |
|  | |
| WindowHandle | [getSystemHandle](http://docs.google.com/classsf_1_1Window.htm#a26368e7162229f8637c34d80ab0f138e) () const |
|  | Get the OS-specific handle of the window. |
|  | |

| Protected Member Functions | |
| --- | --- |
| virtual void | [onCreate](http://docs.google.com/classsf_1_1Window.htm#a106633b9be49b27f83d4712689b493eb) () |
|  | Function called after the window has been created. |
|  | |
| virtual void | [onResize](http://docs.google.com/classsf_1_1Window.htm#a10f567a387da7b49f417f73321fcf64d) () |
|  | Function called after the window has been resized. |
|  | |

| Static Private Member Functions | |
| --- | --- |
| static void | [ensureGlContext](http://docs.google.com/classsf_1_1GlResource.htm#ae0efa7935241644608ca32ba47b22a33) () |
|  | Make sure that a valid OpenGL context exists in the current thread. |
|  | |

## Detailed Description

[Window](http://docs.google.com/classsf_1_1Window.htm) that serves as a target for OpenGL rendering.

[sf::Window](http://docs.google.com/classsf_1_1Window.htm) is the main class of the [Window](http://docs.google.com/classsf_1_1Window.htm) module.

It defines an OS window that is able to receive an OpenGL rendering.

A [sf::Window](http://docs.google.com/classsf_1_1Window.htm) can create its own new window, or be embedded into an already existing control using the create(handle) function. This can be useful for embedding an OpenGL rendering area into a view which is part of a bigger GUI with existing windows, controls, etc. It can also serve as embedding an OpenGL rendering area into a window created by another (probably richer) GUI library like Qt or wxWidgets.

The [sf::Window](http://docs.google.com/classsf_1_1Window.htm) class provides a simple interface for manipulating the window: move, resize, show/hide, control mouse cursor, etc. It also provides event handling through its [pollEvent()](http://docs.google.com/classsf_1_1Window.htm#a338e996585faf82e93069858e3b531b7) and [waitEvent()](http://docs.google.com/classsf_1_1Window.htm#aaf02ab64fbc1d374eef3696df54137bc) functions.

Note that OpenGL experts can pass their own parameters (antialiasing level, bits for the depth and stencil buffers, etc.) to the OpenGL context attached to the window, with the [sf::ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm) structure which is passed as an optional argument when creating the window.

Usage example:

// Declare and create a new window

[sf::Window](http://docs.google.com/classsf_1_1Window.htm) window([sf::VideoMode](http://docs.google.com/classsf_1_1VideoMode.htm)(800, 600), "SFML window");

// Limit the framerate to 60 frames per second (this step is optional)

window.[setFramerateLimit](http://docs.google.com/classsf_1_1Window.htm#af4322d315baf93405bf0d5087ad5e784)(60);

// The main loop - ends as soon as the window is closed

while (window.[isOpen](http://docs.google.com/classsf_1_1Window.htm#a5aa9c2b2b0e51d3423c2b66c80253337)())

{

// Event processing

[sf::Event](http://docs.google.com/classsf_1_1Event.htm) event;

while (window.[pollEvent](http://docs.google.com/classsf_1_1Window.htm#a338e996585faf82e93069858e3b531b7)(event))

{

// Request for closing the window

if (event.[type](http://docs.google.com/classsf_1_1Event.htm#adf2f8044f713fd9d6019077b0d1ffe0a) == [sf::Event::Closed](http://docs.google.com/classsf_1_1Event.htm#af41fa9ed45c02449030699f671331d4aa316e4212e083f1dce79efd8d9e9c0a95))

window.[close](http://docs.google.com/classsf_1_1Window.htm#a99d1e030387b0c26f5995670504fe7b5)();

}

// Activate the window for OpenGL rendering

window.[setActive](http://docs.google.com/classsf_1_1Window.htm#a17ccf8ece0ce0bf2f1e6698bcfa29731)();

// OpenGL drawing commands go here...

// End the current frame and display its contents on screen

window.[display](http://docs.google.com/classsf_1_1Window.htm#adabf839cb103ac96cfc82f781638772a)();

}

Definition at line [57](http://docs.google.com/Window_2Window_8hpp_source.htm#l00057) of file [Window/Window.hpp](http://docs.google.com/Window_2Window_8hpp_source.htm).

## Constructor & Destructor Documentation

| sf::Window::Window | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Default constructor.

This constructor doesn't actually create the window, use the other constructors or call "create" to do so.

| sf::Window::Window | ( | [VideoMode](http://docs.google.com/classsf_1_1VideoMode.htm) | *mode*, |
| --- | --- | --- | --- |
|  |  | const [String](http://docs.google.com/classsf_1_1String.htm) & | *title*, |
|  |  | Uint32 | *style* = Style::Default, |
|  |  | const [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm) & | *settings* = [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm)() |
|  | ) |  |  |

Construct a new window.

This constructor creates the window with the size and pixel depth defined in *mode*. An optional style can be passed to customize the look and behaviour of the window (borders, title bar, resizable, closable, ...). If *style* contains Style::Fullscreen, then *mode* must be a valid video mode.

The fourth parameter is an optional structure specifying advanced OpenGL context settings such as antialiasing, depth-buffer bits, etc.

Parameters

| mode | Video mode to use (defines the width, height and depth of the rendering area of the window) |
| --- | --- |
| title | Title of the window |
| style | [Window](http://docs.google.com/classsf_1_1Window.htm) style |
| settings | Additional settings for the underlying OpenGL context |

| | sf::Window::Window | ( | WindowHandle | *handle*, | | --- | --- | --- | --- | |  |  | const [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm) & | *settings* = [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm)() | |  | ) |  |  | | explicit |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Construct the window from an existing control.

Use this constructor if you want to create an OpenGL rendering area into an already existing control.

The second parameter is an optional structure specifying advanced OpenGL context settings such as antialiasing, depth-buffer bits, etc.

Parameters

| handle | Platform-specific handle of the control |
| --- | --- |
| settings | Additional settings for the underlying OpenGL context |

| | virtual sf::Window::~Window | ( |  | ) |  | | --- | --- | --- | --- | --- | | virtual |
| --- | --- | --- | --- | --- | --- | --- |

Destructor.

Closes the window and free all the resources attached to it.

## Member Function Documentation

| void sf::Window::close | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Close the window and destroy all the attached resources.

After calling this function, the [sf::Window](http://docs.google.com/classsf_1_1Window.htm) instance remains valid and you can call [create()](http://docs.google.com/classsf_1_1Window.htm#a30e6edf2162f8dbff61023b9de5d961d) to recreate the window. All other functions such as [pollEvent()](http://docs.google.com/classsf_1_1Window.htm#a338e996585faf82e93069858e3b531b7) or [display()](http://docs.google.com/classsf_1_1Window.htm#adabf839cb103ac96cfc82f781638772a) will still work (i.e. you don't have to test [isOpen()](http://docs.google.com/classsf_1_1Window.htm#a5aa9c2b2b0e51d3423c2b66c80253337) every time), and will have no effect on closed windows.

| void sf::Window::create | ( | [VideoMode](http://docs.google.com/classsf_1_1VideoMode.htm) | *mode*, |
| --- | --- | --- | --- |
|  |  | const [String](http://docs.google.com/classsf_1_1String.htm) & | *title*, |
|  |  | Uint32 | *style* = Style::Default, |
|  |  | const [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm) & | *settings* = [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm)() |
|  | ) |  |  |

Create (or recreate) the window.

If the window was already created, it closes it first. If *style* contains Style::Fullscreen, then *mode* must be a valid video mode.

Parameters

| mode | Video mode to use (defines the width, height and depth of the rendering area of the window) |
| --- | --- |
| title | Title of the window |
| style | [Window](http://docs.google.com/classsf_1_1Window.htm) style |
| settings | Additional settings for the underlying OpenGL context |

| void sf::Window::create | ( | WindowHandle | *handle*, |
| --- | --- | --- | --- |
|  |  | const [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm) & | *settings* = [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm)() |
|  | ) |  |  |

Create (or recreate) the window from an existing control.

Use this function if you want to create an OpenGL rendering area into an already existing control. If the window was already created, it closes it first.

Parameters

| handle | Platform-specific handle of the control |
| --- | --- |
| settings | Additional settings for the underlying OpenGL context |

| void sf::Window::display | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Display on screen what has been rendered to the window so far.

This function is typically called after all OpenGL rendering has been done for the current frame, in order to show it on screen.

| [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) sf::Window::getPosition | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the position of the window.

ReturnsPosition of the window, in pixels See Also[setPosition](http://docs.google.com/classsf_1_1Window.htm#a6c4078bfbf61c29bfc4b4732ce764f17)

| const [ContextSettings](http://docs.google.com/structsf_1_1ContextSettings.htm)& sf::Window::getSettings | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the settings of the OpenGL context of the window.

Note that these settings may be different from what was passed to the constructor or the [create()](http://docs.google.com/classsf_1_1Window.htm#a30e6edf2162f8dbff61023b9de5d961d) function, if one or more settings were not supported. In this case, SFML chose the closest match.

ReturnsStructure containing the OpenGL context settings

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

Get the size of the rendering region of the window.

The size doesn't include the titlebar and borders of the window.

ReturnsSize in pixels See Also[setSize](http://docs.google.com/classsf_1_1Window.htm#ad6513418bb2963347cd1819a1810524d)

| WindowHandle sf::Window::getSystemHandle | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the OS-specific handle of the window.

The type of the returned handle is sf::WindowHandle, which is a typedef to the handle type defined by the OS. You shouldn't need to use this function, unless you have very specific stuff to implement that SFML doesn't support, or implement a temporary workaround until a bug is fixed.

ReturnsSystem handle of the window

| bool sf::Window::isOpen | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Tell whether or not the window is open.

This function returns whether or not the window exists. Note that a hidden window (setVisible(false)) is open (therefore this function would return true).

ReturnsTrue if the window is open, false if it has been closed

| | virtual void sf::Window::onCreate | ( |  | ) |  | | --- | --- | --- | --- | --- | | protectedvirtual |
| --- | --- | --- | --- | --- | --- | --- |

Function called after the window has been created.

This function is called so that derived classes can perform their own specific initialization as soon as the window is created.

Reimplemented in [sf::RenderWindow](http://docs.google.com/classsf_1_1RenderWindow.htm#a5bef0040b0fa87bed9fbd459c980d53a).

| | virtual void sf::Window::onResize | ( |  | ) |  | | --- | --- | --- | --- | --- | | protectedvirtual |
| --- | --- | --- | --- | --- | --- | --- |

Function called after the window has been resized.

This function is called so that derived classes can perform custom actions when the size of the window changes.

Reimplemented in [sf::RenderWindow](http://docs.google.com/classsf_1_1RenderWindow.htm#a5c85fe482313562d33ffd24a194b6fef).

| bool sf::Window::pollEvent | ( | [Event](http://docs.google.com/classsf_1_1Event.htm) & | *event* | ) |  |
| --- | --- | --- | --- | --- | --- |

Pop the event on top of the event queue, if any, and return it.

This function is not blocking: if there's no pending event then it will return false and leave *event* unmodified. Note that more than one event may be present in the event queue, thus you should always call this function in a loop to make sure that you process every pending event.

[sf::Event](http://docs.google.com/classsf_1_1Event.htm) event;

while (window.[pollEvent](http://docs.google.com/classsf_1_1Window.htm#a338e996585faf82e93069858e3b531b7)(event))

{

// process event...

}

Parameters

| event | [Event](http://docs.google.com/classsf_1_1Event.htm) to be returned |
| --- | --- |

ReturnsTrue if an event was returned, or false if the event queue was empty See Also[waitEvent](http://docs.google.com/classsf_1_1Window.htm#aaf02ab64fbc1d374eef3696df54137bc)

| bool sf::Window::setActive | ( | bool | *active* = true | ) | const |
| --- | --- | --- | --- | --- | --- |

Activate or deactivate the window as the current target for OpenGL rendering.

A window is active only on the current thread, if you want to make it active on another thread you have to deactivate it on the previous thread first if it was active. Only one window can be active on a thread at a time, thus the window previously active (if any) automatically gets deactivated.

Parameters

| active | True to activate, false to deactivate |
| --- | --- |

ReturnsTrue if operation was successful, false otherwise

| void sf::Window::setFramerateLimit | ( | unsigned int | *limit* | ) |  |
| --- | --- | --- | --- | --- | --- |

Limit the framerate to a maximum fixed frequency.

If a limit is set, the window will use a small delay after each call to [display()](http://docs.google.com/classsf_1_1Window.htm#adabf839cb103ac96cfc82f781638772a) to ensure that the current frame lasted long enough to match the framerate limit. SFML will try to match the given limit as much as it can, but since it internally uses sf::sleep, whose precision depends on the underlying OS, the results may be a little unprecise as well (for example, you can get 65 FPS when requesting 60).

Parameters

| limit | Framerate limit, in frames per seconds (use 0 to disable limit) |
| --- | --- |

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

Change the window's icon.

*pixels* must be an array of *width* x *height* pixels in 32-bits RGBA format.

The OS default icon is used by default.

Parameters

| width | Icon's width, in pixels |
| --- | --- |
| height | Icon's height, in pixels |
| pixels | Pointer to the array of pixels in memory |

See Also[setTitle](http://docs.google.com/classsf_1_1Window.htm#a3b3f3513bb6be90f5cd456c20b5fd5fa)

| void sf::Window::setJoystickThreshold | ( | float | *threshold* | ) |  |
| --- | --- | --- | --- | --- | --- |

Change the joystick threshold.

The joystick threshold is the value below which no JoystickMoved event will be generated.

The threshold value is 0.1 by default.

Parameters

| threshold | New threshold, in the range [0, 100] |
| --- | --- |

| void sf::Window::setKeyRepeatEnabled | ( | bool | *enabled* | ) |  |
| --- | --- | --- | --- | --- | --- |

Enable or disable automatic key-repeat.

If key repeat is enabled, you will receive repeated KeyPressed events while keeping a key pressed. If it is disabled, you will only get a single event when the key is pressed.

Key repeat is enabled by default.

Parameters

| enabled | True to enable, false to disable |
| --- | --- |

| void sf::Window::setMouseCursorVisible | ( | bool | *visible* | ) |  |
| --- | --- | --- | --- | --- | --- |

Show or hide the mouse cursor.

The mouse cursor is visible by default.

Parameters

| visible | True to show the mouse cursor, false to hide it |
| --- | --- |

| void sf::Window::setPosition | ( | const [Vector2i](http://docs.google.com/classsf_1_1Vector2.htm) & | *position* | ) |  |
| --- | --- | --- | --- | --- | --- |

Change the position of the window on screen.

This function only works for top-level windows (i.e. it will be ignored for windows created from the handle of a child window/control).

Parameters

| position | New position, in pixels |
| --- | --- |

See Also[getPosition](http://docs.google.com/classsf_1_1Window.htm#a2e6bc12612ea289afea8268fe37c8678)

| void sf::Window::setSize | ( | const [Vector2u](http://docs.google.com/classsf_1_1Vector2.htm) | *size* | ) |  |
| --- | --- | --- | --- | --- | --- |

Change the size of the rendering region of the window.

Parameters

| size | New size, in pixels |
| --- | --- |

See Also[getSize](http://docs.google.com/classsf_1_1Window.htm#ad2b55a731ba1680fe67292991ef1610e)

| void sf::Window::setTitle | ( | const [String](http://docs.google.com/classsf_1_1String.htm) & | *title* | ) |  |
| --- | --- | --- | --- | --- | --- |

Change the title of the window.

Parameters

| title | New title |
| --- | --- |

See Also[setIcon](http://docs.google.com/classsf_1_1Window.htm#a63af61e026fba08e3153fd013620bcc0)

| void sf::Window::setVerticalSyncEnabled | ( | bool | *enabled* | ) |  |
| --- | --- | --- | --- | --- | --- |

Enable or disable vertical synchronization.

Activating vertical synchronization will limit the number of frames displayed to the refresh rate of the monitor. This can avoid some visual artifacts, and limit the framerate to a good value (but not constant across different computers).

Vertical synchronization is disabled by default.

Parameters

| enabled | True to enable v-sync, false to deactivate it |
| --- | --- |

| void sf::Window::setVisible | ( | bool | *visible* | ) |  |
| --- | --- | --- | --- | --- | --- |

Show or hide the window.

The window is shown by default.

Parameters

| visible | True to show the window, false to hide it |
| --- | --- |

| bool sf::Window::waitEvent | ( | [Event](http://docs.google.com/classsf_1_1Event.htm) & | *event* | ) |  |
| --- | --- | --- | --- | --- | --- |

Wait for an event and return it.

This function is blocking: if there's no pending event then it will wait until an event is received. After this function returns (and no error occured), the *event* object is always valid and filled properly. This function is typically used when you have a thread that is dedicated to events handling: you want to make this thread sleep as long as no new event is received.

[sf::Event](http://docs.google.com/classsf_1_1Event.htm) event;

if (window.[waitEvent](http://docs.google.com/classsf_1_1Window.htm#aaf02ab64fbc1d374eef3696df54137bc)(event))

{

// process event...

}

Parameters

| event | [Event](http://docs.google.com/classsf_1_1Event.htm) to be returned |
| --- | --- |

ReturnsFalse if any error occured See Also[pollEvent](http://docs.google.com/classsf_1_1Window.htm#a338e996585faf82e93069858e3b531b7)

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

* [Window/Window.hpp](http://docs.google.com/Window_2Window_8hpp_source.htm)

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