

Deprecated List

Class `sf::Event::MouseWheelEvent`

This event is deprecated and potentially inaccurate. Use `MouseWheelScrolledEvent` instead.

Member `sf::LinesStrip`

Use `LineStrip` instead

Member `sf::RenderWindow::capture () const`

Use a `sf::Texture` and its `sf::Texture::update(const Window&)` function and a `sf::Image` instead.

```
1 sf::Vector2u windowSize = window.getSize();
2 sf::Texture texture;
3 texture.create(windowSize.x, windowSize.y);
4 texture.update(window);
5 sf::Image screenshot = texture.copyToImage();
```

Member `sf::Shader::setParameter (const std::string &name, const`

Use `setUniform(const std::string&, const Gsl::Vec2&)` instead.

Member `sf::Shader::setParameter (const std::string &name, float x,`

Use `setUniform(const std::string&, const Gsl::Vec2&)` instead.

Member `sf::Shader::setParameter (const std::string &name, const`

Use `setUniform(const std::string&, const Gsl::Vec3&)` instead.

Member `sf::Shader::setParameter (const std::string &name, const`

Use `setUniform(const std::string&, const Gsl::Vec4&)` instead.

Member `sf::Shader::setParameter (const std::string &name, const`

Use `setUniform(const std::string&, const Texture&)` instead.

Member `sf::Shader::setParameter (const std::string &name, float x,`

Use `setUniform(const std::string&, const Gsl::Vec3&)` instead.

Member `sf::Shader::setParameter (const std::string &name, const`

Use `setUniform(const std::string&, const Gsl::Mat4&)` instead.

Member `sf::Shader::setParameter (const std::string &name, float x,`

Use `setUniform(const std::string&, const Gsl::Vec4&)` instead.

Member `sf::Shader::setParameter (const std::string &name, float x,`

Use `setUniform(const std::string&, float)` instead.

Member `sf::Shader::setParameter (const std::string &name, Current`

Use `setUniform(const std::string&, CurrentTextureType)` instead.

Member `sf::Text::getColor () const`

There is now fill and outline colors instead of a single global color. Use `getFillColor()` and `getOutlineColor()` instead.

Member `sf::Text::setColor (const Color &color)`

There is now fill and outline colors instead of a single global color. Use `sf::Color` instead.

Member `sf::TrianglesFan`

Use `TriangleFan` instead

Member `sf::TrianglesStrip`

Use `TriangleStrip` instead

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

Modules

[Namespaces](#)

[Classes](#)

[Files](#)

Modules

Here is a list of all modules:

Audio module	Sounds, streaming (musics or custom sources), recording
Graphics module	2D graphics module: sprites, text, shapes, ..
Network module	Socket-based communication, utilities and higher-level
System module	Base module of SFML, defining various utilities
Window module	Provides OpenGL-based windows, and abstractions f

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

Classes

Audio module

Sounds, streaming (musics or custom sources), recording, spatialization.

Classes

class [sf::ALResource](#)

Base class for classes that require an OpenAL context. [More...](#)

class [sf::InputSoundFile](#)

Provide read access to sound files. [More...](#)

class [sf::Listener](#)

The audio listener is the point in the scene from where all the sou

class [sf::Music](#)

Streamed music played from an audio file. [More...](#)

class [sf::OutputSoundFile](#)

Provide write access to sound files. [More...](#)

class [sf::Sound](#)

Regular sound that can be played in the audio environment. [More...](#)

class [sf::SoundBuffer](#)

Storage for audio samples defining a sound. [More...](#)

class [sf::SoundBufferRecorder](#)

Specialized [SoundRecorder](#) which stores the captured audio data

class [sf::SoundFileFactory](#)

Manages and instantiates sound file readers and writers. [More...](#)

class [sf::SoundFileReader](#)

Abstract base class for sound file decoding. [More...](#)

class [sf::SoundFileWriter](#)

Abstract base class for sound file encoding. [More...](#)

class [sf::SoundRecorder](#)

Abstract base class for capturing sound data. [More...](#)

class [sf::SoundSource](#)

Base class defining a sound's properties. [More...](#)

class [sf::SoundStream](#)

Abstract base class for streamed audio sources. [More...](#)

Detailed Description

Sounds, streaming (musics or custom sources), recording, spatialization.

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Protected Member Functions](#) | [List of all members](#)

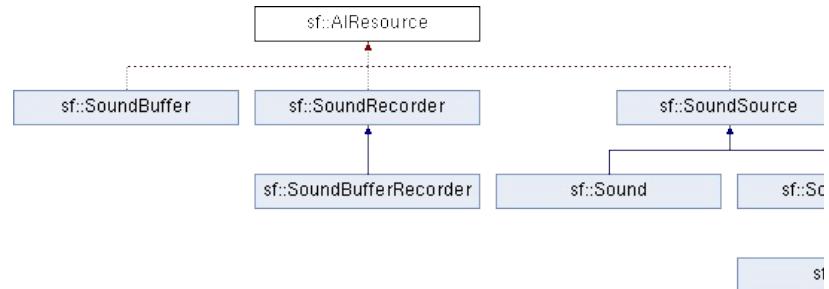
sf::AlResource Class Reference

[Audio module](#)

Base class for classes that require an OpenAL context. More...

```
#include <AlResource.hpp>
```

Inheritance diagram for sf::AlResource:



Protected Member Functions

[AIResource \(\)](#)

Default constructor. More...

[~AIResource \(\)](#)

Destructor. More...

Detailed Description

Base class for classes that require an OpenAL context.

This class is for internal use only, it must be the base of every class that r
order to work.

Definition at line 40 of file [AIResource.hpp](#).

Constructor & Destructor Documentation

sf::AIResource::AIResource ()

Default constructor.

sf::AIResource::~AIResource ()

Destructor.

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

- [AIResource.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

Classes

[Files](#)

[Class List](#)

[Class Index](#)

[Class Hierarchy](#)

[Class Members](#)

[Public Member Functions](#) | [List of all members](#)

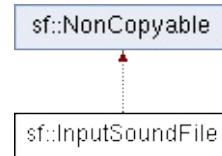
sf::InputSoundFile Class Reference

[Audio module](#)

Provide read access to sound files. More...

```
#include <InputSoundFile.hpp>
```

Inheritance diagram for sf::InputSoundFile:



Public Member Functions

`InputSoundFile ()`
Default constructor. [More...](#)

`~InputSoundFile ()`
Destructor. [More...](#)

`bool openFromFile (const std::string &filename)`
Open a sound file from the disk for reading. [More...](#)

`bool openFromMemory (const void *data, std::size_t sizeInBytes)`
Open a sound file in memory for reading. [More...](#)

`bool openFromStream (InputStream &stream)`
Open a sound file from a custom stream for reading. [More...](#)

`bool openForWriting (const std::string &filename, unsigned int c
sampleRate)`
Open the sound file from the disk for writing. [More...](#)

`Uint64 getSampleCount () const`
Get the total number of audio samples in the file. [More...](#)

`unsigned int getChannelCount () const`
Get the number of channels used by the sound. [More...](#)

`unsigned int getSampleRate () const`
Get the sample rate of the sound. [More...](#)

`Time getDuration () const`
Get the total duration of the sound file. [More...](#)

```
void seek (Uint64 sampleOffset)
```

Change the current read position to the given sample offset.

```
void seek (Time timeOffset)
```

Change the current read position to the given time offset. [More...](#)

```
Uint64 read (Int16 *samples, Uint64 maxCount)
```

Read audio samples from the open file. [More...](#)

Detailed Description

Provide read access to sound files.

This class decodes audio samples from a sound file.

It is used internally by higher-level classes such as `sf::SoundBuffer` and `sf::Music`. If you want to process or analyze audio files without playing them, or if you want a version of `sf::Music` with more specific features.

Usage example:

```
// Open a sound file
sf::InputSoundFile file;
if (!file.openFromFile("music.ogg"))
/* error */;

// Print the sound attributes
std::cout << "duration: " << file.getDuration().asSeconds() << std::endl;
std::cout << "channels: " << file.getChannelCount() << std::endl;
std::cout << "sample rate: " << file.getSampleRate() << std::endl;
std::cout << "sample count: " << file.getSampleCount() << std::endl;

// Read and process batches of samples until the end of file is reached
sf::Int16 samples[1024];
sf::Uint64 count;
do
{
    count = file.read(samples, 1024);
    // process, analyze, play, convert, or whatever
    // you want to do with the samples...
} while (count > 0);
```

See also

`sf::SoundFileReader`, `sf::OutputSoundFile`

Definition at line 46 of file `InputSoundFile.hpp`.

Constructor & Destructor Documentation

sf::InputSoundFile::InputSoundFile ()

Default constructor.

sf::InputSoundFile::~InputSoundFile ()

Destructor.

Member Function Documentation

unsigned int sf::InputSoundFile::getChannelCount() const

Get the number of channels used by the sound.

Returns

Number of channels (1 = mono, 2 = stereo)

Time sf::InputSoundFile::getDuration() const

Get the total duration of the sound file.

This function is provided for convenience, the duration is deduced from the file's metadata.

Returns

Duration of the sound file

Uint64 sf::InputSoundFile::getSampleCount() const

Get the total number of audio samples in the file.

Returns

Number of samples

unsigned int sf::InputSoundFile::getSampleRate () const

Get the sample rate of the sound.

Returns

Sample rate, in samples per second

```
bool sf::InputSoundFile::openForWriting ( const std::string & filename,
                                         unsigned int channelCount,
                                         unsigned int sampleRate
                                       )
```

Open the sound file from the disk for writing.

Parameters

filename Path of the sound file to write

channelCount Number of channels in the sound

sampleRate Sample rate of the sound

Returns

True if the file was successfully opened

```
bool sf::InputSoundFile::openFromFile ( const std::string & filename )
```

Open a sound file from the disk for reading.

The supported audio formats are: WAV (PCM only), OGG/Vorbis, FLAC. FLAC and WAV are 8, 16, 24 and 32 bit.

Parameters

filename Path of the sound file to load

Returns

True if the file was successfully opened

```
bool sf::InputSoundFile::openFromMemory ( const void * data,  
                                         std::size_t sizeInBytes  
                                       )
```

Open a sound file in memory for reading.

The supported audio formats are: WAV (PCM only), OGG/Vorbis, FLAC. FLAC and WAV are 8, 16, 24 and 32 bit.

Parameters

data Pointer to the file data in memory
sizeInBytes Size of the data to load, in bytes

Returns

True if the file was successfully opened

```
bool sf::InputSoundFile::openFromStream ( InputStream & stream )
```

Open a sound file from a custom stream for reading.

The supported audio formats are: WAV (PCM only), OGG/Vorbis, FLAC. FLAC and WAV are 8, 16, 24 and 32 bit.

Parameters

stream Source stream to read from

Returns

True if the file was successfully opened

```
Uint64 sf::InputSoundFile::read ( Int16 * samples,  
                                Uint64 maxCount  
                            )
```

Read audio samples from the open file.

Parameters

samples Pointer to the sample array to fill

maxCount Maximum number of samples to read

Returns

Number of samples actually read (may be less than *maxCount*)

```
void sf::InputSoundFile::seek ( Uint64 sampleOffset )
```

Change the current read position to the given sample offset.

This function takes a sample offset to provide maximum precision. If you use the other overload.

The sample offset takes the channels into account. Offsets can be calculated as $\text{sampleRate} * \text{channelCount}$. If the given offset exceeds the total number of samples in the end of the sound file.

Parameters

sampleOffset Index of the sample to jump to, relative to the beginning of the file.

void sf::InputSoundFile::seek (Time timeOffset)

Change the current read position to the given time offset.

Using a time offset is handy but imprecise. If you need an accurate result, use [seekSample](#) which takes a sample offset.

If the given time exceeds the total duration, this function jumps to the end of the file.

Parameters

timeOffset Time to jump to, relative to the beginning

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

- [InputSoundFile.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#).

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Static Public Member Functions](#) | [List of all members](#)

sf::Listener Class Reference

[Audio module](#)

The audio listener is the point in the scene from where all the sounds are |

```
#include <Listener.hpp>
```

Static Public Member Functions

static void **setGlobalVolume** (float volume)
Change the global volume of all the sounds and musics.

static float **getGlobalVolume** ()
Get the current value of the global volume. [More...](#)

static void **setPosition** (float x, float y, float z)
Set the position of the listener in the scene. [More...](#)

static void **setPosition** (const Vector3f &position)
Set the position of the listener in the scene. [More...](#)

static Vector3f **getPosition** ()
Get the current position of the listener in the scene. [More](#)

static void **setDirection** (float x, float y, float z)
Set the forward vector of the listener in the scene. [More..](#)

static void **setDirection** (const Vector3f &direction)
Set the forward vector of the listener in the scene. [More..](#)

static Vector3f **getDirection** ()
Get the current forward vector of the listener in the scene

static void **setUpVector** (float x, float y, float z)
Set the upward vector of the listener in the scene. [More..](#)

static void **setUpVector** (const Vector3f &upVector)
Set the upward vector of the listener in the scene. [More..](#)

static Vector3f **getUpVector** ()

Get the current upward vector of the listener in the scene

Detailed Description

The audio listener is the point in the scene from where all the sounds are heard.

The audio listener defines the global properties of the audio environment, and musics are heard.

If `sf::View` is the eyes of the user, then `sf::Listener` is his ears (by the way it has the same position, orientation, etc.).

`sf::Listener` is a simple interface, which allows to setup the listener in the scene (position, direction and up vector), and to adjust the global volume.

Because the listener is unique in the scene, `sf::Listener` only contains static methods and cannot be instantiated.

Usage example:

```
// Move the listener to the position (1, 0, -5)
sf::Listener::setPosition(1, 0, -5);

// Make it face the right axis (1, 0, 0)
sf::Listener::setDirection(1, 0, 0);

// Reduce the global volume
sf::Listener::setGlobalVolume(50);
```

Definition at line 42 of file `Listener.hpp`.

Member Function Documentation

static Vector3f sf::Listener::getDirection ()

Get the current forward vector of the listener in the scene.

Returns

Listener's forward vector (not normalized)

See also

[setDirection](#)

static float sf::Listener::getGlobalVolume ()

Get the current value of the global volume.

Returns

Current global volume, in the range [0, 100]

See also

[setGlobalVolume](#)

static Vector3f sf::Listener::getPosition ()

Get the current position of the listener in the scene.

Returns

Listener's position

See also

[setPosition](#)

static **Vector3f** sf::Listener::getUpVector()

Get the current upward vector of the listener in the scene.

Returns

Listener's upward vector (not normalized)

See also

[setUpVector](#)

static void sf::Listener::setDirection(float **x**, float **y**, float **z**)

Set the forward vector of the listener in the scene.

The direction (also called "at vector") is the vector pointing forward. Together with the up vector, it defines the 3D orientation of the listener if it doesn't have to be normalized. The default listener's direction is (0, 0, -1).

Parameters

x X coordinate of the listener's direction

y Y coordinate of the listener's direction

z Z coordinate of the listener's direction

See also

[getDirection](#), [setUpVector](#), [setPosition](#)

static void sf::Listener::setDirection (const Vector3f & direction)

Set the forward vector of the listener in the scene.

The direction (also called "at vector") is the vector pointing forward. Together with the up vector, it defines the 3D orientation of the listener if it doesn't have to be normalized. The default listener's direction is (0, 0, -1).

Parameters

direction New listener's direction

See also

[getDirection](#), [setUpVector](#), [setPosition](#)

static void sf::Listener::setGlobalVolume (float volume)

Change the global volume of all the sounds and musics.

The volume is a number between 0 and 100; it is combined with the individual volume of each sound or music. The default value for the volume is 100 (maximum).

Parameters

volume New global volume, in the range [0, 100]

See also

[getGlobalVolume](#)

```
static void sf::Listener::setPosition ( float x,  
                                     float y,  
                                     float z  
                                 )
```

Set the position of the listener in the scene.

The default listener's position is (0, 0, 0).

Parameters

- x** X coordinate of the listener's position
- y** Y coordinate of the listener's position
- z** Z coordinate of the listener's position

See also

[getPosition](#), [setDirection](#)

```
static void sf::Listener::setPosition ( const Vector3f & position )
```

Set the position of the listener in the scene.

The default listener's position is (0, 0, 0).

Parameters

- position** New listener's position

See also

[getPosition](#), [setDirection](#)

```
static void sf::Listener::setUpVector ( float x,
```

```
    float y,  
    float z  
)
```

Set the upward vector of the listener in the scene.

The up vector is the vector that points upward from the listener's perspective. It defines the 3D orientation of the listener in the scene. The up vector. The default listener's up vector is (0, 1, 0). It is usually not necessary for most scenarios.

Parameters

- x** X coordinate of the listener's up vector
- y** Y coordinate of the listener's up vector
- z** Z coordinate of the listener's up vector

See also

[getUpVector](#), [setDirection](#), [setPosition](#)

static void sf::Listener::setUpVector (const Vector3f & upVector)

Set the upward vector of the listener in the scene.

The up vector is the vector that points upward from the listener's perspective. It defines the 3D orientation of the listener in the scene. The up vector. The default listener's up vector is (0, 1, 0). It is usually not necessary for most scenarios.

Parameters

- upVector** New listener's up vector

See also

`getUpVector`, `setDirection`, `setPosition`

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

- `Listener.hpp`
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [Protected Member Functions](#) | [Protected Attributes](#) | [List of all members](#)

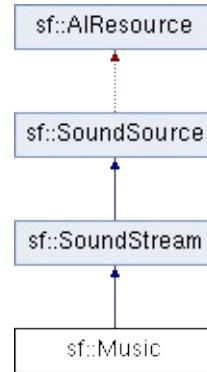
sf::Music Class Reference

[Audio module](#)

Streamed music played from an audio file. More...

```
#include <Music.hpp>
```

Inheritance diagram for sf::Music:



Public Types

enum Status { Stopped, Paused, Playing }

Enumeration of the sound source states. More...

Public Member Functions

`Music ()`

Default constructor. [More...](#)

`~Music ()`

Destructor. [More...](#)

`bool openFromFile (const std::string &filename)`

Open a music from an audio file. [More...](#)

`bool openFromMemory (const void *data, std::size_t sizeInBytes)`

Open a music from an audio file in memory. [More...](#)

`bool openFromStream (InputStream &stream)`

Open a music from an audio file in a custom stream. [More...](#)

`Time getDuration () const`

Get the total duration of the music. [More...](#)

`void play ()`

Start or resume playing the audio stream. [More...](#)

`void pause ()`

Pause the audio stream. [More...](#)

`void stop ()`

Stop playing the audio stream. [More...](#)

`unsigned int getChannelCount () const`

Return the number of channels of the stream. [More...](#)

`unsigned int getSampleRate () const`

Get the stream sample rate of the stream. [More...](#)

Status `getStatus () const`

Get the current status of the stream (stopped, paused, play

void `setPlayingOffset (Time timeOffset)`

Change the current playing position of the stream. [More...](#)

Time `getPlayingOffset () const`

Get the current playing position of the stream. [More...](#)

void `setLoop (bool loop)`

Set whether or not the stream should loop after reaching th

bool `getLoop () const`

Tell whether or not the stream is in loop mode. [More...](#)

void `setPitch (float pitch)`

Set the pitch of the sound. [More...](#)

void `setVolume (float volume)`

Set the volume of the sound. [More...](#)

void `setPosition (float x, float y, float z)`

Set the 3D position of the sound in the audio scene. [More..](#)

void `setPosition (const Vector3f &position)`

Set the 3D position of the sound in the audio scene. [More..](#)

void `setRelativeToListener (bool relative)`

Make the sound's position relative to the listener or absolut

void `setMinDistance (float distance)`

Set the minimum distance of the sound. [More...](#)

void setAttenuation (float attenuation)
Set the attenuation factor of the sound. [More...](#)

float getPitch () const
Get the pitch of the sound. [More...](#)

float getVolume () const
Get the volume of the sound. [More...](#)

Vector3f getPosition () const
Get the 3D position of the sound in the audio scene. [More..](#)

bool isRelativeToListener () const
Tell whether the sound's position is relative to the listener or absolute.

float getMinDistance () const
Get the minimum distance of the sound. [More...](#)

float getAttenuation () const
Get the attenuation factor of the sound. [More...](#)

Protected Member Functions

virtual bool `onGetData (Chunk &data)`

Request a new chunk of audio samples from the stream source.

virtual void `onSeek (Time timeOffset)`

Change the current playing position in the stream source. [More...](#)

void `initialize (unsigned int channelCount, unsigned int sampleRate)`

Define the audio stream parameters. [More...](#)

Protected Attributes

unsigned int `m_source`

OpenAL source identifier. [More...](#)

Detailed Description

Streamed music played from an audio file.

Musics are sounds that are streamed rather than completely loaded in memory.

This is especially useful for compressed musics that usually take longer to load uncompressed: by streaming it instead of loading it entirely, you avoid almost no loading delay. This implies that the underlying resource (file) remains valid for the lifetime of the `sf::Music` object.

Apart from that, a `sf::Music` has almost the same features as the `sf::Source`: you can play/pause/stop it, request its parameters (channels, sample rate), change its position, volume, 3D position, etc.

As a sound stream, a music is played in its own thread in order not to block the main thread. This means that you can leave the music alone after calling `play()`, it will manage itself.

Usage example:

```
// Declare a new music
sf::Music music;

// Open it from an audio file
if (!music.openFromFile("music.ogg"))
{
    // error...
}

// Change some parameters
music.setPosition(0, 1, 10); // change its 3D position
music.setPitch(2);          // increase the pitch
music.setVolume(50);         // reduce the volume
music.setLoop(true);         // make it loop

// Play it
music.play();
```

See also

`sf::Sound`, `sf::SoundStream`

Definition at line 48 of file `Music.hpp`.

Member Enumeration Documentation

enum sf::SoundSource::Status

Enumeration of the sound source states.

Enumerator	
Stopped	Sound is not playing.
Paused	Sound is paused.
Playing	Sound is playing.

Definition at line 50 of file SoundSource.hpp.

Constructor & Destructor Documentation

sf::Music::Music ()

Default constructor.

sf::Music::~Music ()

Destructor.

Member Function Documentation

float sf::SoundSource::getAttenuation() const

Get the attenuation factor of the sound.

Returns

Attenuation factor of the sound

See also

[setAttenuation](#), [getMinDistance](#)

unsigned int sf::SoundStream::getChannelCount() const

Return the number of channels of the stream.

1 channel means a mono sound, 2 means stereo, etc.

Returns

Number of channels

Time sf::Music::getDuration() const

Get the total duration of the music.

Returns

Music duration

bool sf::SoundStream::getLoop () const

Tell whether or not the stream is in loop mode.

Returns

True if the stream is looping, false otherwise

See also

[setLoop](#)

float sf::SoundSource::getMinDistance () const

Get the minimum distance of the sound.

Returns

Minimum distance of the sound

See also

[setMinDistance](#), [getAttenuation](#)

float sf::SoundSource::getPitch () const

Get the pitch of the sound.

Returns

Pitch of the sound

See also

`setPitch`

Time sf::SoundStream::getPlayingOffset() const

Get the current playing position of the stream.

Returns

Current playing position, from the beginning of the stream

See also

`setPlayingOffset`

Vector3f sf::SoundSource::getPosition() const

Get the 3D position of the sound in the audio scene.

Returns

Position of the sound

See also

`setPosition`

unsigned int sf::SoundStream::getSampleRate() const

Get the stream sample rate of the stream.

The sample rate is the number of audio samples played per second. The

Returns

Sample rate, in number of samples per second

Status sf::SoundStream::getStatus() const

Get the current status of the stream (stopped, paused, playing)

Returns

Current status

float sf::SoundSource::getVolume() const

Get the volume of the sound.

Returns

Volume of the sound, in the range [0, 100]

See also

[setVolume](#)

void sf::SoundStream::initialize(unsigned int channelCount, unsigned int sampleRate)

Define the audio stream parameters.

This function must be called by derived classes as soon as they know the play. Any attempt to manipulate the stream ([play\(\)](#), ...) before calling this multiple times if the settings of the audio stream change, but only when t

Parameters

channelCount Number of channels of the stream
sampleRate Sample rate, in samples per second

bool sf::SoundSource::isRelativeToListener () const

Tell whether the sound's position is relative to the listener or is absolute.

Returns

True if the position is relative, false if it's absolute

See also

[setRelativeToListener](#)

virtual bool sf::Music::onGetData (Chunk & data)

Request a new chunk of audio samples from the stream source.

This function fills the chunk from the next samples to read from the audi

Parameters

data Chunk of data to fill

Returns

True to continue playback, false to stop

Implements [sf::SoundStream](#).

virtual void sf::Music::onSeek (Time timeOffset)

Change the current playing position in the stream source.

Parameters

timeOffset New playing position, from the beginning of the music

Implements [sf::SoundStream](#).

bool sf::Music::openFromFile (const std::string & filename)

Open a music from an audio file.

This function doesn't start playing the music (call [play\(\)](#) to do so). See [sf::InputSoundFile](#) for the list of supported formats.

Warning

Since the music is not loaded at once but rather streamed continuously, it is accessible until the [sf::Music](#) object loads a new music or is destroyed.

Parameters

filename Path of the music file to open

Returns

True if loading succeeded, false if it failed

See also

[openFromMemory](#), [openFromStream](#)

bool sf::Music::openFromMemory (const void * data, std::size_t sizeInBytes)

Open a music from an audio file in memory.

This function doesn't start playing the music (call `play()` to do so). See `sf::InputSoundFile` for the list of supported formats.

Warning

Since the music is not loaded at once but rather streamed continuously, it is accessible until the `sf::Music` object loads a new music or is destroyed. It is recommended to destroy the buffer right after calling this function.

Parameters

data Pointer to the file data in memory
sizeInBytes Size of the data to load, in bytes

Returns

True if loading succeeded, false if it failed

See also

`openFromFile`, `openFromStream`

bool sf::Music::openFromStream (`InputStream & stream`)

Open a music from an audio file in a custom stream.

This function doesn't start playing the music (call `play()` to do so). See `sf::InputSoundFile` for the list of supported formats.

Warning

Since the music is not loaded at once but rather streamed continuously, it is accessible until the `sf::Music` object loads a new music or is destroyed. It is recommended to destroy the buffer right after calling this function.

Parameters

stream Source stream to read from

Returns

True if loading succeeded, false if it failed

See also

[openFromFile](#), [openFromMemory](#)

void sf::SoundStream::pause ()

Pause the audio stream.

This function pauses the stream if it was playing, otherwise (stream already stopped).

See also

[play](#), [stop](#)

void sf::SoundStream::play ()

Start or resume playing the audio stream.

This function starts the stream if it was stopped, resumes it if it was paused or beginning if it was already playing. This function uses its own thread so the program while the stream is played.

See also

[pause](#), [stop](#)

void sf::SoundSource::setAttenuation (float attenuation)

Set the attenuation factor of the sound.

The attenuation is a multiplicative factor which makes the sound more or less loud depending on its distance from the listener. An attenuation of 0 will produce a non-attenuated sound, which always be the same whether it is heard from near or from far. On the other hand, an attenuation value such as 100 will make the sound fade out very quickly as it gets further away. The maximum value of the attenuation is 1.

Parameters

attenuation New attenuation factor of the sound

See also

[getAttenuation](#), [setMinDistance](#)

void sf::SoundStream::setLoop (bool loop)

Set whether or not the stream should loop after reaching the end.

If set, the stream will restart from beginning after reaching the end and looping until `setLoop(false)` is called. The default looping state for streams is false.

Parameters

loop True to play in loop, false to play once

See also

[getLoop](#)

void sf::SoundSource::setMinDistance (float distance)

Set the minimum distance of the sound.

The "minimum distance" of a sound is the maximum distance at which it can still be heard. Further than the minimum distance, it will start to fade out according to its attenuation.

("inside the head of the listener") is an invalid value and is forbidden. The distance is 1.

Parameters

distance New minimum distance of the sound

See also

[getMinDistance](#), [setAttenuation](#)

void sf::SoundSource::setPitch (float pitch)

Set the pitch of the sound.

The pitch represents the perceived fundamental frequency of a sound; it can be acute or grave by changing its pitch. A side effect of changing the pitch is to change the sound as well. The default value for the pitch is 1.

Parameters

pitch New pitch to apply to the sound

See also

[getPitch](#)

void sf::SoundStream::setPlayingOffset (Time timeOffset)

Change the current playing position of the stream.

The playing position can be changed when the stream is either paused or stopped. The playing position when the stream is stopped has no effect, since playing the stream must be explicitly started again.

Parameters

timeOffset New playing position, from the beginning of the stream

See also

[getPlayingOffset](#)

```
void sf::SoundSource::setPosition ( float x,  
                                  float y,  
                                  float z  
 )
```

Set the 3D position of the sound in the audio scene.

Only sounds with one channel (mono sounds) can be spatialized. The default value is (0, 0, 0).

Parameters

x X coordinate of the position of the sound in the scene

y Y coordinate of the position of the sound in the scene

z Z coordinate of the position of the sound in the scene

See also

[getPosition](#)

```
void sf::SoundSource::setPosition ( const Vector3f & position )
```

Set the 3D position of the sound in the audio scene.

Only sounds with one channel (mono sounds) can be spatialized. The default value is (0, 0, 0).

Parameters

position Position of the sound in the scene

See also

[getPosition](#)

void sf::SoundSource::setRelativeToListener (bool relative)

Make the sound's position relative to the listener or absolute.

Making a sound relative to the listener will ensure that it will always be |
of the position of the listener. This can be useful for non-spatialized soun
the listener, or sounds attached to it. The default value is false (position i

Parameters

relative True to set the position relative, false to set it absolute

See also

[isRelativeToListener](#)

void sf::SoundSource::setVolume (float volume)

Set the volume of the sound.

The volume is a value between 0 (mute) and 100 (full volume). The defa

Parameters

volume Volume of the sound

See also

[getVolume](#)

void sf::SoundStream::stop ()

Stop playing the audio stream.

This function stops the stream if it was playing or paused, and does not also resets the playing position (unlike [pause\(\)](#)).

See also

[play](#), [pause](#)

Member Data Documentation

unsigned int sf::SoundSource::m_source

OpenAL source identifier.

Definition at line 274 of file SoundSource.hpp.

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

- Music.hpp
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

Classes

[Files](#)

[Class List](#)

[Class Index](#)

[Class Hierarchy](#)

[Class Members](#)

[Public Member Functions](#) | [List of all members](#)

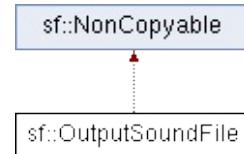
sf::OutputSoundFile Class Reference

[Audio module](#)

Provide write access to sound files. More...

```
#include <OutputSoundFile.hpp>
```

Inheritance diagram for sf::OutputSoundFile:



Public Member Functions

[OutputSoundFile \(\)](#)

Default constructor. [More...](#)

[~OutputSoundFile \(\)](#)

Destructor. [More...](#)

bool [openFromFile \(const std::string &filename, unsigned int sampleRate\)](#)
Open the sound file from the disk for writing. [More...](#)

void [write \(const Int16 *samples, Uint64 count\)](#)

Write audio samples to the file. [More...](#)

Detailed Description

Provide write access to sound files.

This class encodes audio samples to a sound file.

It is used internally by higher-level classes such as `sf::SoundBuffer`, but create audio files from custom data sources, like generated audio samples:

Usage example:

```
// Create a sound file, ogg/vorbis format, 44100 Hz, stereo
sf::OutputSoundFile file;
if (!file.openFromFile("music.ogg", 44100, 2))
/* error */;

while (...)

{
    // Read or generate audio samples from your custom source
    std::vector<sf::Int16> samples = ...;

    // Write them to the file
    file.write(samples.data(), samples.size());
}
```

See also

`sf::SoundFileWriter`, `sf::InputSoundFile`

Definition at line 44 of file `OutputSoundFile.hpp`.

Constructor & Destructor Documentation

sf::OutputSoundFile::OutputSoundFile ()

Default constructor.

sf::OutputSoundFile::~OutputSoundFile ()

Destructor.

Closes the file if it was still open.

Member Function Documentation

```
bool sf::OutputSoundFile::openFromFile ( const std::string & filename,
                                         unsigned int sampleRate,
                                         unsigned int channelCount
                                         )
```

Open the sound file from the disk for writing.

The supported audio formats are: WAV, OGG/Vorbis, FLAC.

Parameters

filename Path of the sound file to write
sampleRate Sample rate of the sound
channelCount Number of channels in the sound

Returns

True if the file was successfully opened

```
void sf::OutputSoundFile::write ( const Int16 * samples,
                                 Uint64 count
                                 )
```

Write audio samples to the file.

Parameters

samples Pointer to the sample array to write
count Number of samples to write

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

- [OutputSoundFile.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [Protected Attributes](#) | [List of all members](#)

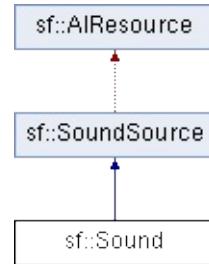
sf::Sound Class Reference

[Audio module](#)

Regular sound that can be played in the audio environment. [More...](#)

```
#include <Sound.hpp>
```

Inheritance diagram for sf::Sound:



Public Types

enum Status { Stopped, Paused, Playing }

Enumeration of the sound source states. More...

Public Member Functions

`Sound ()`

Default constructor. [More...](#)

`Sound (const SoundBuffer &buffer)`

Construct the sound with a buffer. [More...](#)

`Sound (const Sound ©)`

Copy constructor. [More...](#)

`~Sound ()`

Destructor. [More...](#)

`void play ()`

Start or resume playing the sound. [More...](#)

`void pause ()`

Pause the sound. [More...](#)

`void stop ()`

stop playing the sound [More...](#)

`void setBuffer (const SoundBuffer &buffer)`

Set the source buffer containing the audio data to p

`void setLoop (bool loop)`

Set whether or not the sound should loop after read

`void setPlayingOffset (Time timeOffset)`

Change the current playing position of the sound. [More...](#)

`const SoundBuffer * getBuffer () const`

Get the audio buffer attached to the sound. [More...](#)

`bool getLoop () const`

Tell whether or not the sound is in loop mode. [More...](#)

`Time getPlayingOffset () const`

Get the current playing position of the sound. [More...](#)

`Status getStatus () const`

Get the current status of the sound (stopped, paus

`Sound & operator= (const Sound &right)`

Overload of assignment operator. [More...](#)

`void resetBuffer ()`

Reset the internal buffer of the sound. [More...](#)

`void setPitch (float pitch)`

Set the pitch of the sound. [More...](#)

`void setVolume (float volume)`

Set the volume of the sound. [More...](#)

`void setPosition (float x, float y, float z)`

Set the 3D position of the sound in the audio scene

`void setPosition (const Vector3f &position)`

Set the 3D position of the sound in the audio scene

`void setRelativeToListener (bool relative)`

Make the sound's position relative to the listener or

`void setMinDistance (float distance)`

Set the minimum distance of the sound. [More...](#)

`void setAttenuation (float attenuation)`

Set the attenuation factor of the sound. More...

`float getPitch () const`

Get the pitch of the sound. More...

`float getVolume () const`

Get the volume of the sound. More...

`Vector3f getPosition () const`

Get the 3D position of the sound in the audio scene.

`bool isRelativeToListener () const`

Tell whether the sound's position is relative to the listener.

`float getMinDistance () const`

Get the minimum distance of the sound. More...

`float getAttenuation () const`

Get the attenuation factor of the sound. More...

Protected Attributes

unsigned int `m_source`

OpenAL source identifier. [More...](#)

Detailed Description

Regular sound that can be played in the audio environment.

`sf::Sound` is the class to use to play sounds.

It provides:

- Control (play, pause, stop)
- Ability to modify output parameters in real-time (pitch, volume, ...)
- 3D spatial features (position, attenuation, ...).

`sf::Sound` is perfect for playing short sounds that can fit in memory and re-use them (like gun shots). For longer sounds, like background musics or long speeches, it's better to use `sf::Music` based on streaming).

In order to work, a sound must be given a buffer of audio data to play. This is done by creating a `sf::SoundBuffer`, and attaching it to a sound with the `setBuffer()` function. Note that multiple buffers can't be attached to the same sound at the same time.

Usage example:

```
sf::SoundBuffer buffer;
buffer.loadFromFile("sound.wav");

sf::Sound sound;
sound.setBuffer(buffer);
sound.play();
```

See also

`sf::SoundBuffer`, `sf::Music`

Definition at line 45 of file Sound.hpp.

Member Enumeration Documentation

enum sf::SoundSource::Status

Enumeration of the sound source states.

Enumerator	
Stopped	Sound is not playing.
Paused	Sound is paused.
Playing	Sound is playing.

Definition at line 50 of file SoundSource.hpp.

Constructor & Destructor Documentation

`sf::Sound::Sound()`

Default constructor.

`sf::Sound::Sound(const SoundBuffer & buffer)`

Construct the sound with a buffer.

Parameters

buffer `Sound` buffer containing the audio data to play with the sound.

`sf::Sound::Sound(const Sound & copy)`

Copy constructor.

Parameters

copy Instance to copy

`sf::Sound::~Sound()`

Destructor.

Member Function Documentation

float sf::SoundSource::getAttenuation() const

Get the attenuation factor of the sound.

Returns

Attenuation factor of the sound

See also

[setAttenuation](#), [getMinDistance](#)

const SoundBuffer* sf::Sound::getBuffer() const

Get the audio buffer attached to the sound.

Returns

[Sound](#) buffer attached to the sound (can be NULL)

bool sf::Sound::getLoop() const

Tell whether or not the sound is in loop mode.

Returns

True if the sound is looping, false otherwise

See also

[setLoop](#)

float sf::SoundSource::getMinDistance() const

Get the minimum distance of the sound.

Returns

Minimum distance of the sound

See also

[setMinDistance](#), [getAttenuation](#)

float sf::SoundSource::getPitch() const

Get the pitch of the sound.

Returns

Pitch of the sound

See also

[setPitch](#)

Time sf::Sound::getPlayingOffset() const

Get the current playing position of the sound.

Returns

Current playing position, from the beginning of the sound

See also

[setPlayingOffset](#)

Vector3f sf::SoundSource::getPosition () const

Get the 3D position of the sound in the audio scene.

Returns

Position of the sound

See also

[setPosition](#)

Status sf::Sound::getStatus () const

Get the current status of the sound (stopped, paused, playing)

Returns

Current status of the sound

float sf::SoundSource::getVolume () const

Get the volume of the sound.

Returns

Volume of the sound, in the range [0, 100]

See also

[setVolume](#)

bool sf::SoundSource::isRelativeToListener() const

Tell whether the sound's position is relative to the listener or is absolute.

Returns

True if the position is relative, false if it's absolute

See also

[setRelativeToListener](#)

Sound& sf::Sound::operator=(const Sound & right)

Overload of assignment operator.

Parameters

right Instance to assign

Returns

Reference to self

void sf::Sound::pause()

Pause the sound.

This function pauses the sound if it was playing, otherwise (sound already stopped).

See also

[play](#), [stop](#)

void sf::Sound::play ()

Start or resume playing the sound.

This function starts the stream if it was stopped, resumes it if it was at the beginning if it was it already playing. This function uses its own thread so the program while the sound is played.

See also

[pause](#), [stop](#)

void sf::Sound::resetBuffer ()

Reset the internal buffer of the sound.

This function is for internal use only, you don't have to use it. It is called by the sound uses, when it is destroyed in order to prevent the sound from using memory.

void sf::SoundSource::setAttenuation (float attenuation)

Set the attenuation factor of the sound.

The attenuation is a multiplicative factor which makes the sound more or less audible over distance from the listener. An attenuation of 0 will produce a non-attenuated sound, which always be the same whether it is heard from near or from far. On the other hand, a value such as 100 will make the sound fade out very quickly as it gets further away. The maximum value of the attenuation is 1.

Parameters

attenuation New attenuation factor of the sound

See also

[getAttenuation](#), [setMinDistance](#)

void sf::Sound::setBuffer (const SoundBuffer & buffer)

Set the source buffer containing the audio data to play.

It is important to note that the sound buffer is not copied, thus the `sf::Sound` object will stay alive as long as it is attached to the sound.

Parameters

buffer Sound buffer to attach to the sound

See also

[getBuffer](#)

void sf::Sound::setLoop (bool loop)

Set whether or not the sound should loop after reaching the end.

If set, the sound will restart from beginning after reaching the end and `setLoop(false)` is called. The default looping state for sound is false.

Parameters

loop True to play in loop, false to play once

See also

[getLoop](#)

void sf::SoundSource::setMinDistance (float **distance)**

Set the minimum distance of the sound.

The "minimum distance" of a sound is the maximum distance at which it is heard. Further than the minimum distance, it will start to fade out according to its attenuation profile. A negative value or zero ("inside the head of the listener") is an invalid value and is forbidden. The default value for the minimum distance is 1.

Parameters

distance New minimum distance of the sound

See also

[getMinDistance](#), [setAttenuation](#)

void sf::SoundSource::setPitch (float **pitch)**

Set the pitch of the sound.

The pitch represents the perceived fundamental frequency of a sound; that is, it can make a sound acute or grave by changing its pitch. A side effect of changing the pitch is that it changes the volume of the sound as well. The default value for the pitch is 1.

Parameters

pitch New pitch to apply to the sound

See also

[getPitch](#)

void sf::Sound::setPlayingOffset (Time timeOffset)

Change the current playing position of the sound.

The playing position can be changed when the sound is either paused or playing. The position when the sound is stopped has no effect, since playing the sound starts from the beginning again.

Parameters

timeOffset New playing position, from the beginning of the sound

See also

[getPlayingOffset](#)

void sf::SoundSource::setPosition (float x, float y, float z)

Set the 3D position of the sound in the audio scene.

Only sounds with one channel (mono sounds) can be spatialized. The default value is (0, 0, 0).

Parameters

x X coordinate of the position of the sound in the scene

y Y coordinate of the position of the sound in the scene

z Z coordinate of the position of the sound in the scene

See also

[getPosition](#)

void sf::SoundSource::setPosition (const Vector3f & position)

Set the 3D position of the sound in the audio scene.

Only sounds with one channel (mono sounds) can be spatialized. The default value is (0, 0, 0).

Parameters

position Position of the sound in the scene

See also

[getPosition](#)

void sf::SoundSource::setRelativeToListener (bool relative)

Make the sound's position relative to the listener or absolute.

Making a sound relative to the listener will ensure that it will always be positioned relative to the position of the listener. This can be useful for non-spatialized sounds that are attached to the listener, or sounds attached to it. The default value is false (position is absolute).

Parameters

relative True to set the position relative, false to set it absolute

See also

[isRelativeToListener](#)

void sf::SoundSource::setVolume (float volume)

Set the volume of the sound.

The volume is a value between 0 (mute) and 100 (full volume). The default value is 100.

Parameters

volume Volume of the sound

See also

[getVolume](#)

void sf::Sound::stop ()

stop playing the sound

This function stops the sound if it was playing or paused, and does not also resets the playing position (unlike [pause\(\)](#)).

See also

[play](#), [pause](#)

Member Data Documentation

unsigned int sf::SoundSource::m_source

OpenAL source identifier.

Definition at line 274 of file SoundSource.hpp.

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

- [Sound.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [Friends](#) | [List of all members](#)

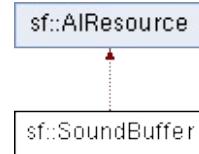
sf::SoundBuffer Class Reference

[Audio module](#)

Storage for audio samples defining a sound. [More...](#)

```
#include <SoundBuffer.hpp>
```

Inheritance diagram for sf::SoundBuffer:



Public Member Functions

`SoundBuffer ()`
Default constructor. [More...](#)

`SoundBuffer (const SoundBuffer ©)`
Copy constructor. [More...](#)

`~SoundBuffer ()`
Destructor. [More...](#)

`bool loadFromFile (const std::string &filename)`
Load the sound buffer from a file. [More...](#)

`bool loadFromMemory (const void *data, std::size_t sizeInBytes)`
Load the sound buffer from a file in memory. [More...](#)

`bool loadFromStream (InputStream &stream)`
Load the sound buffer from a custom stream. [More...](#)

`bool loadFromSamples (const Int16 *samples, Uint64 sampleCount, unsigned int sampleRate)`
Load the sound buffer from an array of audio samples. [More...](#)

`bool saveToFile (const std::string &filename) const`
Save the sound buffer to an audio file. [More...](#)

`const Int16 * getSamples () const`
Get the array of audio samples stored in the buffer. [More...](#)

`Uint64 getSampleCount () const`
Get the number of samples stored in the buffer. [More...](#)

`unsigned int getSampleRate () const`
Get the sample rate of the sound. [More...](#)

`unsigned int getChannelCount () const`
Get the number of channels used by the sound. [More...](#)

`Time getDuration () const`
Get the total duration of the sound. [More...](#)

`SoundBuffer & operator= (const SoundBuffer &right)`
Overload of assignment operator. [More...](#)

Friends

class **Sound**

Detailed Description

Storage for audio samples defining a sound.

A sound buffer holds the data of a sound, which is an array of audio samples.

A sample is a 16 bits signed integer that defines the amplitude of the sound. It is then reconstituted by playing these samples at a high rate (for example, standard rate used for playing CDs). In short, audio samples are like texture pixels. A sound buffer is similar to a [sf::Texture](#).

A sound buffer can be loaded from a file (see [loadFromFile\(\)](#)) for the convenience, or from memory, from a custom stream (see [sf::InputStream](#)) or directly from another sound buffer. It can also be saved back to a file.

Sound buffers alone are not very useful: they hold the audio data but cannot play it. To do so, you have to use the [sf::Sound](#) class, which provides functions to play/pause/stop the sound and to change the way it is outputted (volume, pitch, 3D position, ...). This separation allows for better performances: indeed a [sf::SoundBuffer](#) is a heavy resource, and any operation on it is slow (especially for real-time applications). On the other side, a [sf::Sound](#) is a lightweight object that holds a reference to a sound buffer and change the way it is played without actually reading it. It is also possible to bind several [sf::Sound](#) instances to the same [sf::SoundBuffer](#).

It is important to note that the [sf::Sound](#) instance doesn't copy the buffer, but only holds a reference to it. Thus, a [sf::SoundBuffer](#) must not be destructed while it is used by a [sf::Sound](#). For example, if you write a function that uses a local [sf::SoundBuffer](#) instance for loading a sound, you must make sure that the [sf::Sound](#) instance still exists when the function returns.

Usage example:

```
// Declare a new sound buffer
sf::SoundBuffer buffer;
```

```
// Load it from a file
if (!buffer.loadFromFile("sound.wav"))
{
    // error...
}

// Create a sound source and bind it to the buffer
sf::Sound sound1;
sound1.setBuffer(buffer);

// Play the sound
sound1.play();

// Create another sound source bound to the same buffer
sf::Sound sound2;
sound2.setBuffer(buffer);

// Play it with a higher pitch -- the first sound remains unchanged
sound2.setPitch(2);
sound2.play();
```

See also

[sf::Sound](#), [sf::SoundBufferRecorder](#)

Definition at line 49 of file [SoundBuffer.hpp](#).

Constructor & Destructor Documentation

sf::SoundBuffer::SoundBuffer()

Default constructor.

sf::SoundBuffer::SoundBuffer(const SoundBuffer & *copy*)

Copy constructor.

Parameters

copy Instance to copy

sf::SoundBuffer::~SoundBuffer()

Destructor.

Member Function Documentation

unsigned int sf::SoundBuffer::getChannelCount() const

Get the number of channels used by the sound.

If the sound is mono then the number of channels will be 1, 2 for stereo,

Returns

Number of channels

See also

[getSampleRate](#), [getDuration](#)

Time sf::SoundBuffer::getDuration() const

Get the total duration of the sound.

Returns

[Sound](#) duration

See also

[getSampleRate](#), [getChannelCount](#)

Uint64 sf::SoundBuffer::getSampleCount() const

Get the number of samples stored in the buffer.

The array of samples can be accessed with the [getSamples\(\)](#) function.

Returns

Number of samples

See also

[getSamples](#)

unsigned int sf::SoundBuffer::getSampleRate () const

Get the sample rate of the sound.

The sample rate is the number of samples played per second. The example, 44100 samples/s is CD quality).

Returns

Sample rate (number of samples per second)

See also

[getChannelCount](#), [getDuration](#)

const Int16* sf::SoundBuffer::getSamples () const

Get the array of audio samples stored in the buffer.

The format of the returned samples is 16 bits signed integer (sf::Int16). this array is given by the [getSampleCount\(\)](#) function.

Returns

Read-only pointer to the array of sound samples

See also

[getSampleCount](#)

bool sf::SoundBuffer::loadFromFile (const std::string & filename)

Load the sound buffer from a file.

See the documentation of [sf::InputSoundFile](#) for the list of supported formats.

Parameters

filename Path of the sound file to load

Returns

True if loading succeeded, false if it failed

See also

[loadFromMemory](#), [loadFromStream](#), [loadFromSamples](#), [saveToFile](#)

bool sf::SoundBuffer::loadFromMemory (const void * data, std::size_t sizeInBytes)

Load the sound buffer from a file in memory.

See the documentation of [sf::InputSoundFile](#) for the list of supported formats.

Parameters

data Pointer to the file data in memory

sizeInBytes Size of the data to load, in bytes

Returns

True if loading succeeded, false if it failed

See also

[loadFromFile](#), [loadFromStream](#), [loadFromSamples](#)

```
bool sf::SoundBuffer::loadFromSamples ( const Int16 * samples,  
                                      UInt64      sampleCount,  
                                      unsigned int channelCount,  
                                      unsigned int sampleRate  
                                    )
```

Load the sound buffer from an array of audio samples.

The assumed format of the audio samples is 16 bits signed integer (sf::Int16).

Parameters

samples Pointer to the array of samples in memory

sampleCount Number of samples in the array

channelCount Number of channels (1 = mono, 2 = stereo, ...)

sampleRate Sample rate (number of samples to play per second)

Returns

True if loading succeeded, false if it failed

See also

[loadFromFile](#), [loadFromMemory](#), [saveToFile](#)

```
bool sf::SoundBuffer::loadFromStream ( InputStream & stream )
```

Load the sound buffer from a custom stream.

See the documentation of `sf::InputSoundFile` for the list of supported formats.

Parameters

stream Source stream to read from

Returns

True if loading succeeded, false if it failed

See also

`loadFromFile`, `loadFromMemory`, `loadFromSamples`

SoundBuffer& sf::SoundBuffer::operator= (const SoundBuffer & right)

Overload of assignment operator.

Parameters

right Instance to assign

Returns

Reference to self

bool sf::SoundBuffer::saveToFile (const std::string & filename) const

Save the sound buffer to an audio file.

See the documentation of `sf::OutputSoundFile` for the list of supported formats.

Parameters

filename Path of the sound file to write

Returns

True if saving succeeded, false if it failed

See also

[loadFromFile](#), [loadFromMemory](#), [loadFromSamples](#)

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

- [SoundBuffer.hpp](#)

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Static Public Member Functions](#) | [Protected Member Functions](#) | [List of all members](#)

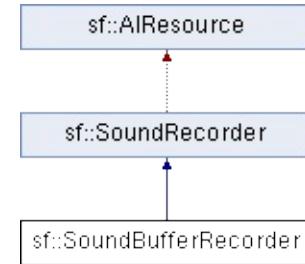
sf::SoundBufferRecorder Class Reference

[Audio module](#)

Specialized [SoundRecorder](#) which stores the captured audio data into a s

```
#include <SoundBufferRecorder.hpp>
```

Inheritance diagram for sf::SoundBufferRecorder:



Public Member Functions

`~SoundBufferRecorder ()`
destructor [More...](#)

`const SoundBuffer & getBuffer () const`
Get the sound buffer containing the captured audio.

`bool start (unsigned int sampleRate=44100)`
Start the capture. [More...](#)

`void stop ()`
Stop the capture. [More...](#)

`unsigned int getSampleRate () const`
Get the sample rate. [More...](#)

`bool setDevice (const std::string &name)`
Set the audio capture device. [More...](#)

`const std::string & getDevice () const`
Get the name of the current audio capture device.

`void setChannelCount (unsigned int channelCount)`
Set the channel count of the audio capture device.

`unsigned int getChannelCount () const`
Get the number of channels used by this recorder.

Static Public Member Functions

```
static std::vector< std::string > getAvailableDevices ()  
    Get a list of the names of all available audio devices.
```

```
static std::string getDefaultDevice ()  
    Get the name of the default audio capture device.
```

```
static bool isAvailable ()  
    Check if the system supports audio capture.
```

Protected Member Functions

virtual bool [onStart \(\)](#)
Start capturing audio data. [More...](#)

virtual bool [onProcessSamples \(const Int16 *samples, std::size_t sampleCount\)](#)
Process a new chunk of recorded samples. [More...](#)

virtual void [onStop \(\)](#)
Stop capturing audio data. [More...](#)

void [setProcessingInterval \(Time interval\)](#)
Set the processing interval. [More...](#)

Detailed Description

Specialized `SoundRecorder` which stores the captured audio data into a `sf::SoundBuffer`.

`sf::SoundBufferRecorder` allows to access a recorded sound through a `sf::SoundBuffer` object (played, saved to a file, etc.).

It has the same simple interface as its base class (`start()`, `stop()`) and returns a `sf::SoundBuffer` containing the recorded sound buffer (`getBuffer()`).

As usual, don't forget to call the `isAvailable()` function before using this class (see the [SoundRecorder](#) page for more details about this).

Usage example:

```
if (sf::SoundBufferRecorder::isAvailable())
{
    // Record some audio data
    sf::SoundBufferRecorder recorder;
    recorder.start();
    ...
    recorder.stop();

    // Get the buffer containing the captured audio data
    const sf::SoundBuffer& buffer = recorder.getBuffer();

    // Save it to a file (for example...)
    buffer.saveToFile("my_record.ogg");
}
```

See also

`sf::SoundRecorder`

Definition at line 44 of file `SoundBufferRecorder.hpp`.

Constructor & Destructor Documentation

sf::SoundBufferRecorder::~SoundBufferRecorder()

destructor

Member Function Documentation

static std::vector<std::string> sf::SoundRecorder::getAvailableDevi

Get a list of the names of all available audio capture devices.

This function returns a vector of strings, containing the names of all avail

Returns

A vector of strings containing the names

const SoundBuffer& sf::SoundBufferRecorder::getBuffer() const

Get the sound buffer containing the captured audio data.

The sound buffer is valid only after the capture has ended. This function returns a reference to the internal sound buffer, but it can be copied if you need to make any modifications.

Returns

Read-only access to the sound buffer

unsigned int sf::SoundRecorder::getChannelCount() const

Get the number of channels used by this recorder.

Currently only mono and stereo are supported, so the value is either 1 (for

Returns

Number of channels

See also

[setChannelCount](#)

static std::string sf::SoundRecorder::getDefaultDevice ()

Get the name of the default audio capture device.

This function returns the name of the default audio capture device. If none is returned.

Returns

The name of the default audio capture device

const std::string& sf::SoundRecorder::getDevice () const

Get the name of the current audio capture device.

Returns

The name of the current audio capture device

unsigned int sf::SoundRecorder::getSampleRate () const

Get the sample rate.

The sample rate defines the number of audio samples captured per second. Higher quality (for example, 44100 samples/sec is CD quality).

Returns

Sample rate, in samples per second

static bool sf::SoundRecorder::isAvailable ()

Check if the system supports audio capture.

This function should always be called before using the audio capture feature. Attempting to use `sf::SoundRecorder` or one of its derived classes will fail.

Returns

True if audio capture is supported, false otherwise

virtual bool sf::SoundBufferRecorder::onProcessSamples

(const Int32 &
std::size_t
)

Process a new chunk of recorded samples.

Parameters

samples Pointer to the new chunk of recorded samples

sampleCount Number of samples pointed by *samples*

Returns

True to continue the capture, or false to stop it

Implements `sf::SoundRecorder`.

virtual bool sf::SoundBufferRecorder::onStart ()

Start capturing audio data.

Returns

True to start the capture, or false to abort it

Reimplemented from `sf::SoundRecorder`.

virtual void sf::SoundBufferRecorder::onStop ()

Stop capturing audio data.

Reimplemented from `sf::SoundRecorder`.

void sf::SoundRecorder::setChannelCount (unsigned int channelC

Set the channel count of the audio capture device.

This method allows you to specify the number of channels used for recording. Mono and 16-bit stereo are supported.

Parameters

channelCount Number of channels. Currently only mono (1) and st

See also

`getChannelCount`

bool sf::SoundRecorder::setDevice (const std::string & name)

Set the audio capture device.

This function sets the audio capture device to the device with the given (i.e: while recording). If you do so while recording and opening the device

Parameters

name The name of the audio capture device

Returns

True, if it was able to set the requested device

See also

[getAvailableDevices](#), [getDefaultDevice](#)

void sf::SoundRecorder::setProcessingInterval (Time interval)

Set the processing interval.

The processing interval controls the period between calls to the onProcess() event. If you want to use a small interval if you want to process the recorded data in real time.

Note: this is only a hint, the actual period may vary. So don't rely on this for precise timing.

The default processing interval is 100 ms.

Parameters

interval Processing interval

bool sf::SoundRecorder::start (unsigned int sampleRate = 44100)

Start the capture.

The `sampleRate` parameter defines the number of audio samples captured per second. The higher the sample rate, the better the quality (for example, 44100 samples/sec is CD quality). This method returns `true` if it was successful. Please note that it doesn't block the rest of the program while the capture runs. Please note that the capture device will be used by default if none was selected before. You can select which capture device will be used by calling the `setDevice()` method. If none was selected before, the default capture device will be used. You can get a list of the names of all available capture devices by calling the `getAvailableDevices()` method.

Parameters

sampleRate Desired capture rate, in number of samples per second

Returns

True, if start of capture was successful

See also

`stop`, `getAvailableDevices`

void sf::SoundRecorder::stop ()

Stop the capture.

See also

`start`

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

- `SoundBufferRecorder.hpp`

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Classes](#) | [Static Public Member Functions](#) | [List of all members](#)

sf::SoundFileFactory Class Reference

[Audio module](#)

Manages and instantiates sound file readers and writers. [More...](#)

```
#include <SoundFileFactory.hpp>
```

Static Public Member Functions

template<typename T >
 static void **registerReader** ()
 Register a new reader. [More...](#)

template<typename T >
 static void **unregisterReader** ()
 Unregister a reader. [More...](#)

template<typename T >
 static void **registerWriter** ()
 Register a new writer. [More...](#)

template<typename T >
 static void **unregisterWriter** ()
 Unregister a writer. [More...](#)

static **SoundFileReader** * **createReaderFromFilename** (const std::string &filename)
 Instantiate the right reader for the given file on disk.

static **SoundFileReader** * **createReaderFromMemory** (const void *data, size_t size)
 Instantiate the right codec for the given file in memory.

static **SoundFileReader** * **createReaderFromStream** (InputStream &stream)
 Instantiate the right codec for the given file in a stream.

static **Sound FileWriter** * **createWriterFromFilename** (const std::string &filename)
 Instantiate the right writer for the given file on disk.

Detailed Description

Manages and instantiates sound file readers and writers.

This class is where all the sound file readers and writers are registered.

You should normally only need to use its registration and unregistration 1 and manipulation are wrapped into the higher-level classes [sf::InputSoundFile](#) and [sf::OutputSoundFile](#).

To register a new reader (writer) use the [sf::SoundFileFactory::registerReader](#) ([sf::SoundFileFactory::registerWriter](#)) function. You don't have to call the [unregisterReader](#) ([unregisterWriter](#)) function to unregister a format before your application ends (typically, when a plugin is unloaded).

Usage example:

```
sf::SoundFileFactory::registerReader<MySoundFileReader>();
sf::SoundFileFactory::registerWriter<MySoundFileWriter>();
```

See also

[sf::InputSoundFile](#), [sf::OutputSoundFile](#), [sf::SoundFileReader](#), [sf::SoundFileWriter](#)

Definition at line 46 of file [SoundFileFactory.hpp](#).

Member Function Documentation

static SoundFileReader*
sf::SoundFileFactory::createReaderFromFilename (con

Instantiate the right reader for the given file on disk.

It's up to the caller to release the returned reader

Parameters

filename Path of the sound file

Returns

A new sound file reader that can read the given file, or null if no read

See also

[createReaderFromMemory](#), [createReaderFromStream](#)

static SoundFileReader*
sf::SoundFileFactory::createReaderFromMemory

Instantiate the right codec for the given file in memory.

It's up to the caller to release the returned reader

Parameters

data Pointer to the file data in memory
sizeInBytes Total size of the file data, in bytes

Returns

A new sound file codec that can read the given file, or null if no code

See also

[createReaderFromFilename](#), [createReaderFromStream](#)

static SoundFileReader*
sf::SoundFileFactory::createReaderFromStream

Instantiate the right codec for the given file in stream.

It's up to the caller to release the returned reader

Parameters

stream Source stream to read from

Returns

A new sound file codec that can read the given file, or null if no code

See also

[createReaderFromFilename](#), [createReaderFromMemory](#)

static SoundFileWriter*
sf::SoundFileFactory::createWriterFromFilename (con

Instantiate the right writer for the given file on disk.

It's up to the caller to release the returned writer

Parameters

filename Path of the sound file

Returns

A new sound file writer that can write given file, or null if no writer ca

template<typename T >

static void sf::SoundFileFactory::registerReader ()

Register a new reader.

See also

[unregisterReader](#)

template<typename T >

static void sf::SoundFileFactory::registerWriter ()

Register a new writer.

See also

[unregisterWriter](#)

template<typename T >

static void sf::SoundFileFactory::unregisterReader ()

Unregister a reader.

See also

[registerReader](#)

```
template<typename T >  
static void sf::SoundFileFactory::unregisterWriter( )
```

Unregister a writer.

See also

[registerWriter](#)

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

- [SoundFileFactory.hpp](#)

SFML is licensed under the terms and conditions of the zlib/png license.
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Classes](#) | [Public Member Functions](#) | [List of all members](#)

sf::SoundFileReader Class Reference abstract

[Audio module](#)

Abstract base class for sound file decoding. [More...](#)

```
#include <SoundFileReader.hpp>
```

Classes

struct [Info](#)

Structure holding the audio properties of a sound file. [More...](#)

Public Member Functions

virtual ~SoundFileReader ()
Virtual destructor. [More...](#)

virtual bool open (InputStream &stream, Info &info)=0
Open a sound file for reading. [More...](#)

virtual void seek (Uint64 sampleOffset)=0
Change the current read position to the given sample offset.

virtual Uint64 read (Int16 *samples, Uint64 maxCount)=0
Read audio samples from the open file. [More...](#)

Detailed Description

Abstract base class for sound file decoding.

This class allows users to read audio file formats not natively supported by SFML.
A list of supported readable audio formats.

A valid sound file reader must override the open, seek and write functions, and implement the check function; the latter is used by SFML to find a suitable writer for a given file.

To register a new reader, use the `sf::SoundFileFactory::registerReader` function.

Usage example:

```
class MySoundFileReader : public sf::SoundFileReader
{
public:
    static bool check(sf::InputStream& stream)
    {
        // typically, read the first few header bytes and check fields that identify
        // return true if the reader can handle the format
    }

    virtual bool open(sf::InputStream& stream, Info& info)
    {
        // read the sound file header and fill the sound attributes
        // (channel count, sample count and sample rate)
        // return true on success
    }

    virtual void seek(sf::Uint64 sampleOffset)
    {
        // advance to the sampleOffset-th sample from the beginning of the sound file
    }

    virtual sf::Uint64 read(sf::Int16* samples, sf::Uint64 maxCount)
    {
        // read up to 'maxCount' samples into the 'samples' array,
        // convert them (for example from normalized float) if they are not stored
        // as 16-bits signed integers in the file
        // return the actual number of samples read
    }
}
```

```
    }
};

sf::SoundFileFactory::registerReader<MySoundFileReader>();
```

See also

[sf::InputSoundFile](#), [sf::SoundFileFactory](#), [sf::SoundFileWriter](#)

Definition at line 43 of file [SoundFileReader.hpp](#).

Constructor & Destructor Documentation

virtual sf::SoundFileReader::~SoundFileReader()

Virtual destructor.

Definition at line 62 of file [SoundFileReader.hpp](#).

Member Function Documentation

```
virtual bool sf::SoundFileReader::open ( InputStream & stream,  
                                         Info & info  
                                         )
```

Open a sound file for reading.

The provided stream reference is valid as long as the SoundFileReader it during the whole lifetime of the reader.

Parameters

stream Source stream to read from

info Structure to fill with the properties of the loaded sound

Returns

True if the file was successfully opened

```
virtual Uint64 sf::SoundFileReader::read ( Int16 * samples,  
                                         Uint64 maxCount  
                                         )
```

Read audio samples from the open file.

Parameters

samples Pointer to the sample array to fill

maxCount Maximum number of samples to read

Returns

Number of samples actually read (may be less than *maxCount*)

virtual void sf::SoundFileReader::seek (UInt64 sampleOffset)

Change the current read position to the given sample offset.

The sample offset takes the channels into account. Offsets can be calculated as $\text{sampleRate} * \text{channelCount}$. If the given offset exceeds the total number of samples in the file, it wraps around to the end of the file.

Parameters

sampleOffset Index of the sample to jump to, relative to the beginning of the file.

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

- [SoundFileReader.hpp](#)

SFML is licensed under the terms and conditions of the [zlib/png](#) license.

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#).

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Attributes](#) | [List of all members](#)

sf::SoundFileReader::Info Struct Reference

Structure holding the audio properties of a sound file. [More...](#)

```
#include <SoundFileReader.hpp>
```

Public Attributes

Uint64 `sampleCount`

Total number of samples in the file. [More...](#)

unsigned int `channelCount`

Number of channels of the sound. [More...](#)

unsigned int `sampleRate`

Samples rate of the sound, in samples per second. [More...](#)

Detailed Description

Structure holding the audio properties of a sound file.

Definition at line 51 of file [SoundFileReader.hpp](#).

Member Data Documentation

unsigned int sf::SoundFileReader::Info::channelCount

Number of channels of the sound.

Definition at line 54 of file [SoundFileReader.hpp](#).

Uint64 sf::SoundFileReader::Info::sampleCount

Total number of samples in the file.

Definition at line 53 of file [SoundFileReader.hpp](#).

unsigned int sf::SoundFileReader::Info::sampleRate

Samples rate of the sound, in samples per second.

Definition at line 55 of file [SoundFileReader.hpp](#).

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

- [SoundFileReader.hpp](#)
-

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [List of all members](#)

sf::SoundFileWriter Class Reference abstract

[Audio module](#)

Abstract base class for sound file encoding. [More...](#)

```
#include <SoundFileWriter.hpp>
```

Public Member Functions

virtual ~[SoundFileWriter](#) ()
Virtual destructor. [More...](#)

virtual bool [open](#) (const std::string &filename, unsigned int sampleRate,
Open a sound file for writing. [More...](#)

virtual void [write](#) (const Int16 *samples, Uint64 count)=0
Write audio samples to the open file. [More...](#)

Detailed Description

Abstract base class for sound file encoding.

This class allows users to write audio file formats not natively supported by SFML or supported writable audio formats.

A valid sound file writer must override the open and write functions, as well as the check function; the latter is used by SFML to find a suitable writer for a given file extension.

To register a new writer, use the `sf::SoundFileFactory::registerWriter` temporary function.

Usage example:

```
class MySoundFileWriter : public sf::SoundFileWriter
{
public:
    static bool check(const std::string& filename)
    {
        // typically, check the extension
        // return true if the writer can handle the format
    }

    virtual bool open(const std::string& filename, unsigned int sampleRate)
    {
        // open the file 'filename' for writing,
        // write the given sample rate and channel count to the file header
        // return true on success
    }

    virtual void write(const sf::Int16* samples, sf::Uint64 count)
    {
        // write 'count' samples stored at address 'samples',
        // convert them (for example to normalized float) if the format requires it
    }
};

sf::SoundFileFactory::registerWriter<MySoundFileWriter>();
```

See also

[sf::OutputSoundFile](#), [sf::SoundFileFactory](#), [sf::SoundFileReader](#)

Definition at line 41 of file [SoundFileWriter.hpp](#).

Constructor & Destructor Documentation

virtual sf::SoundFileWriter::~SoundFileWriter()

Virtual destructor.

Definition at line 49 of file [SoundFileWriter.hpp](#).

Member Function Documentation

```
virtual bool sf::SoundFileWriter::open ( const std::string & filename,
                                         unsigned int sampleRate,
                                         unsigned int channelCount
                                       )
```

Open a sound file for writing.

Parameters

filename Path of the file to open
sampleRate Sample rate of the sound
channelCount Number of channels of the sound

Returns

True if the file was successfully opened

```
virtual void sf::SoundFileWriter::write ( const Int16 * samples,
                                         Uint64 count
                                       )
```

Write audio samples to the open file.

Parameters

samples Pointer to the sample array to write
count Number of samples to write

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

- [SoundFileWriter.hpp](#)

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Static Public Member Functions](#) | [Protected Member Functions](#) | [List of all members](#)

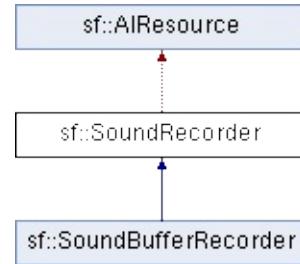
sf::SoundRecorder Class Reference abstract

[Audio module](#)

Abstract base class for capturing sound data. [More...](#)

```
#include <SoundRecorder.hpp>
```

Inheritance diagram for sf::SoundRecorder:



Public Member Functions

virtual ~SoundRecorder ()
destructor [More...](#)

bool [start](#) (unsigned int sampleRate=44100)
Start the capture. [More...](#)

void [stop](#) ()
Stop the capture. [More...](#)

unsigned int [getSampleRate](#) () const
Get the sample rate. [More...](#)

bool [setDevice](#) (const std::string &name)
Set the audio capture device. [More...](#)

const std::string & [getDevice](#) () const
Get the name of the current audio capture device. [More...](#)

void [setChannelCount](#) (unsigned int channelCount)
Set the channel count of the audio capture device. [More...](#)

unsigned int [getChannelCount](#) () const
Get the number of channels used by this recorder. [More...](#)

Static Public Member Functions

```
static std::vector< std::string > getAvailableDevices ()  
    Get a list of the names of all available audio devices.
```

```
static std::string getDefaultDevice ()  
    Get the name of the default audio capture device.
```

```
static bool isAvailable ()  
    Check if the system supports audio capture.
```

Protected Member Functions

`SoundRecorder ()`

Default constructor. [More...](#)

`void setProcessingInterval (Time interval)`

Set the processing interval. [More...](#)

`virtual bool onStart ()`

Start capturing audio data. [More...](#)

`virtual bool onProcessSamples (const Int16 *samples, std::size_t sampl`

Process a new chunk of recorded samples. [More...](#)

`virtual void onStop ()`

Stop capturing audio data. [More...](#)

Detailed Description

Abstract base class for capturing sound data.

`sf::SoundBuffer` provides a simple interface to access the audio recording (microphone).

As an abstract base class, it only cares about capturing sound sample. What is useful with them is left to the derived class. Note that SFML provides a buffer to capture data to a sound buffer (see `sf::SoundBufferRecorder`).

A derived class has only one virtual function to override:

- `onProcessSamples` provides the new chunks of audio samples while the capture is active.

Moreover, two additional virtual functions can be overridden as well if necessary:

- `onStart` is called before the capture happens, to perform custom initialization.
- `onStop` is called after the capture ends, to perform custom cleanup.

A derived class can also control the frequency of the onProcessSamples protected function. The default interval is chosen to avoid to consume too much CPU, but it can be changed to a smaller value if you require more samples in real time, for example.

The audio capture feature may not be supported or activated on every platform. You can check its availability with the `isAvailable()` function. If it returns false, the `SoundBufferRecorder` will fail.

If you have multiple sound input devices connected to your computer (for example, soundcard, webcam mic, ...) you can get a list of all available devices to choose from.

function. You can then select a device by calling `setDevice()` with the `ez` default capturing device will be used.

By default the recording is in 16-bit mono. Using the `setChannelCount` method of channels used by the audio capture device to record. Note that you have to record in mono or stereo before starting the recording.

It is important to note that the audio capture happens in a separate thread of the program. In particular, the `onProcessSamples` virtual function (but not `onStart`) is called from this separate thread. It is important to keep this in mind, because it can cause synchronization issues if you share data between threads. Another thing to remember is to call `stop()` in the destructor of your derived class, so that the recording thread is destroyed.

Usage example:

```
class CustomRecorder : public sf::SoundRecorder
{
    ~CustomRecorder()
    {
        // Make sure to stop the recording thread
        stop();
    }

    virtual bool onStart() // optional
    {
        // Initialize whatever has to be done before the capture starts
        ...

        // Return true to start playing
        return true;
    }

    virtual bool onProcessSamples(const Int16* samples, std::size_t sampleCount)
    {
        // Do something with the new chunk of samples (store them, send them,
        ...

        // Return true to continue playing
        return true;
    }

    virtual void onStop() // optional
    {
        // Clean up whatever has to be done after the capture ends
    }
}
```

```
        ...
    }

// Usage
if (CustomRecorder::isAvailable())
{
    CustomRecorder recorder;

    if (!recorder.start())
        return -1;

    ...
    recorder.stop();
}
```

See also

[sf::SoundBufferRecorder](#)

Definition at line 45 of file [SoundRecorder.hpp](#).

Constructor & Destructor Documentation

virtual sf::SoundRecorder::~SoundRecorder()

destructor

sf::SoundRecorder::SoundRecorder()

Default constructor.

This constructor is only meant to be called by derived classes.

Member Function Documentation

static std::vector<std::string> sf::SoundRecorder::getAvailableDevi

Get a list of the names of all available audio capture devices.

This function returns a vector of strings, containing the names of all avail

Returns

A vector of strings containing the names

unsigned int sf::SoundRecorder::getChannelCount () const

Get the number of channels used by this recorder.

Currently only mono and stereo are supported, so the value is either 1 (fo

Returns

Number of channels

See also

[setChannelCount](#)

static std::string sf::SoundRecorder::getDefaultDevice ()

Get the name of the default audio capture device.

This function returns the name of the default audio capture device. If none is available, an empty string is returned.

Returns

The name of the default audio capture device

const std::string& sf::SoundRecorder::getDevice() const

Get the name of the current audio capture device.

Returns

The name of the current audio capture device

unsigned int sf::SoundRecorder::getSampleRate() const

Get the sample rate.

The sample rate defines the number of audio samples captured per second. Higher sample rates result in better sound quality (for example, 44100 samples/sec is CD quality).

Returns

Sample rate, in samples per second

static bool sf::SoundRecorder::isAvailable()

Check if the system supports audio capture.

This function should always be called before using the audio capture features. If it returns false, any attempt to use `sf::SoundRecorder` or one of its derived classes will fail.

Returns

True if audio capture is supported, false otherwise

```
virtual bool sf::SoundRecorder::onProcessSamples ( const Int16 * samples,
                                                std::size_t sampleCount )

```

Process a new chunk of recorded samples.

This virtual function is called every time a new chunk of recorded data is then do whatever it wants with it (storing it, playing it, sending it over the

Parameters

samples Pointer to the new chunk of recorded samples
sampleCount Number of samples pointed by *samples*

Returns

True to continue the capture, or false to stop it

Implemented in `sf::SoundBufferRecorder`.

```
virtual bool sf::SoundRecorder::onStart ( )
```

Start capturing audio data.

This virtual function may be overridden by a derived class if something happens when capture starts. If not, this function can be ignored; the default implementation returns true.

Returns

True to start the capture, or false to abort it

Reimplemented in [sf::SoundBufferRecorder](#).

virtual void sf::SoundRecorder::onStop ()

Stop capturing audio data.

This virtual function may be overridden by a derived class if something capture ends. If not, this function can be ignored; the default implementa

Reimplemented in [sf::SoundBufferRecorder](#).

void sf::SoundRecorder::setChannelCount (unsigned int channelC

Set the channel count of the audio capture device.

This method allows you to specify the number of channels used for recoi and 16-bit stereo are supported.

Parameters

channelCount Number of channels. Currently only mono (1) and st

See also

[getChannelCount](#)

bool sf::SoundRecorder::setDevice (const std::string & name)

Set the audio capture device.

This function sets the audio capture device to the device with the given

(i.e: while recording). If you do so while recording and opening the device

Parameters

name The name of the audio capture device

Returns

True, if it was able to set the requested device

See also

[getAvailableDevices](#), [getDefaultDevice](#)

void sf::SoundRecorder::setProcessingInterval (Time interval)

Set the processing interval.

The processing interval controls the period between calls to the `onProcess` event. If you want to use a small interval if you want to process the recorded data in real time.

Note: this is only a hint, the actual period may vary. So don't rely on this timing.

The default processing interval is 100 ms.

Parameters

interval Processing interval

bool sf::SoundRecorder::start (unsigned int sampleRate = 44100)

Start the capture.

The `sampleRate` parameter defines the number of audio samples captured per second.

better the quality (for example, 44100 samples/sec is CD quality). This that it doesn't block the rest of the program while the capture runs. Pleas happen at the same time. You can select which capture device will be u setDevice() method. If none was selected before, the default capture de list of the names of all available capture devices by calling `getAvailableD`

Parameters

sampleRate Desired capture rate, in number of samples per second

Returns

True, if start of capture was successful

See also

`stop`, `getAvailableDevices`

`void sf::SoundRecorder::stop()`

Stop the capture.

See also

`start`

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

- `SoundRecorder.hpp`

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [Protected Member Functions](#) | [Protected Attributes](#) | [List of all members](#)

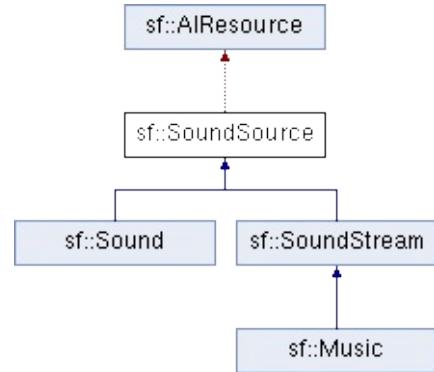
sf::SoundSource Class Reference

[Audio module](#)

Base class defining a sound's properties. More...

```
#include <SoundSource.hpp>
```

Inheritance diagram for sf::SoundSource:



Public Types

enum Status { Stopped, Paused, Playing }

Enumeration of the sound source states. More...

Public Member Functions

`SoundSource (const SoundSource ©)`
Copy constructor. [More...](#)

`virtual ~SoundSource ()`
Destructor. [More...](#)

`void setPitch (float pitch)`
Set the pitch of the sound. [More...](#)

`void setVolume (float volume)`
Set the volume of the sound. [More...](#)

`void setPosition (float x, float y, float z)`
Set the 3D position of the sound in the audio scene. [More...](#)

`void setPosition (const Vector3f &position)`
Set the 3D position of the sound in the audio scene. [More...](#)

`void setRelativeToListener (bool relative)`
Make the sound's position relative to the listener or abs

`void setMinDistance (float distance)`
Set the minimum distance of the sound. [More...](#)

`void setAttenuation (float attenuation)`
Set the attenuation factor of the sound. [More...](#)

`float getPitch () const`
Get the pitch of the sound. [More...](#)

`float getVolume () const`

Get the volume of the sound. [More...](#)

`Vector3f getPosition () const`

Get the 3D position of the sound in the audio scene. [More...](#)

`bool isRelativeToListener () const`

Tell whether the sound's position is relative to the listener. [More...](#)

`float getMinDistance () const`

Get the minimum distance of the sound. [More...](#)

`float getAttenuation () const`

Get the attenuation factor of the sound. [More...](#)

`SoundSource & operator= (const SoundSource &right)`

Overload of assignment operator. [More...](#)

Protected Member Functions

`SoundSource ()`

Default constructor. More...

`Status getStatus () const`

Get the current status of the sound (stopped, paused, playing) [M](#)

Protected Attributes

unsigned int `m_source`

OpenAL source identifier. [More...](#)

Detailed Description

Base class defining a sound's properties.

`sf::SoundSource` is not meant to be used directly, it only serves as a component that can live in the audio environment.

It defines several properties for the sound: pitch, volume, position, attenuation, etc. These properties can be changed at any time with no impact on performances.

See also

`sf::Sound`, `sf::SoundStream`

Definition at line 42 of file `SoundSource.hpp`.

Member Enumeration Documentation

enum sf::SoundSource::Status

Enumeration of the sound source states.

Enumerator	
Stopped	Sound is not playing.
Paused	Sound is paused.
Playing	Sound is playing.

Definition at line 50 of file SoundSource.hpp.

Constructor & Destructor Documentation

sf::SoundSource::SoundSource (const SoundSource & copy)

Copy constructor.

Parameters

copy Instance to copy

virtual sf::SoundSource::~SoundSource ()

Destructor.

sf::SoundSource::SoundSource ()

Default constructor.

This constructor is meant to be called by derived classes only.

Member Function Documentation

float sf::SoundSource::getAttenuation() const

Get the attenuation factor of the sound.

Returns

Attenuation factor of the sound

See also

[setAttenuation](#), [getMinDistance](#)

float sf::SoundSource::getMinDistance() const

Get the minimum distance of the sound.

Returns

Minimum distance of the sound

See also

[setMinDistance](#), [getAttenuation](#)

float sf::SoundSource::getPitch() const

Get the pitch of the sound.

Returns

Pitch of the sound

See also

[setPitch](#)

Vector3f sf::SoundSource::getPosition () const

Get the 3D position of the sound in the audio scene.

Returns

Position of the sound

See also

[setPosition](#)

Status sf::SoundSource::getStatus () const

Get the current status of the sound (stopped, paused, playing)

Returns

Current status of the sound

float sf::SoundSource::getVolume () const

Get the volume of the sound.

Returns

Volume of the sound, in the range [0, 100]

See also

[setVolume](#)

bool sf::SoundSource::isRelativeToListener() const

Tell whether the sound's position is relative to the listener or is absolute.

Returns

True if the position is relative, false if it's absolute

See also

[setRelativeToListener](#)

SoundSource& sf::SoundSource::operator=(const SoundSource &

Overload of assignment operator.

Parameters

right Instance to assign

Returns

Reference to self

void sf::SoundSource::setAttenuation(float attenuation)

Set the attenuation factor of the sound.

The attenuation is a multiplicative factor which makes the sound more or less audible depending on the distance from the listener. An attenuation of 0 will produce a non-attenuated sound, while higher values will reduce its volume at greater distances.

always be the same whether it is heard from near or from far. On the contrary, a value such as 100 will make the sound fade out very quickly as it gets further away. The minimum distance is 0.0f and the maximum distance is infinity. The default value of the attenuation is 1.

Parameters

attenuation New attenuation factor of the sound

See also

[getAttenuation](#), [setMinDistance](#)

void sf::SoundSource::setMinDistance (float distance)

Set the minimum distance of the sound.

The "minimum distance" of a sound is the maximum distance at which it is perceived as being "inside the head of the listener". Further than the minimum distance, it will start to fade out according to its attenuation. A negative value or a value greater than infinity ("inside the head of the listener") is an invalid value and is forbidden. The minimum distance is 0.0f and the maximum distance is infinity. The default minimum distance is 1.

Parameters

distance New minimum distance of the sound

See also

[getMinDistance](#), [setAttenuation](#)

void sf::SoundSource::setPitch (float pitch)

Set the pitch of the sound.

The pitch represents the perceived fundamental frequency of a sound; the higher the pitch, the more acute or grave by changing its pitch. A side effect of changing the pitch is that the volume of the sound may change.

the sound as well. The default value for the pitch is 1.

Parameters

pitch New pitch to apply to the sound

See also

[getPitch](#)

```
void sf::SoundSource::setPosition ( float x,  
                                  float y,  
                                  float z  
    )
```

Set the 3D position of the sound in the audio scene.

Only sounds with one channel (mono sounds) can be spatialized. The default value is (0, 0, 0).

Parameters

x X coordinate of the position of the sound in the scene
y Y coordinate of the position of the sound in the scene
z Z coordinate of the position of the sound in the scene

See also

[getPosition](#)

```
void sf::SoundSource::setPosition ( const Vector3f & position )
```

Set the 3D position of the sound in the audio scene.

Only sounds with one channel (mono sounds) can be spatialized. The default value is (0, 0, 0).

0).

Parameters

position Position of the sound in the scene

See also

[getPosition](#)

void sf::SoundSource::setRelativeToListener (bool **relative**)

Make the sound's position relative to the listener or absolute.

Making a sound relative to the listener will ensure that it will always be of the position of the listener. This can be useful for non-spatialized sounds the listener, or sounds attached to it. The default value is false (position is

Parameters

relative True to set the position relative, false to set it absolute

See also

[isRelativeToListener](#)

void sf::SoundSource::setVolume (float **volume**)

Set the volume of the sound.

The volume is a value between 0 (mute) and 100 (full volume). The default

Parameters

volume Volume of the sound

See also

[getVolume](#)

Member Data Documentation

unsigned int sf::SoundSource::m_source

OpenAL source identifier.

Definition at line 274 of file SoundSource.hpp.

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

- [SoundSource.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Classes](#) | [Public Types](#) | [Public Member Functions](#) | [Protected Member Functions](#) | [Protected Attributes](#) | [List of all members](#)

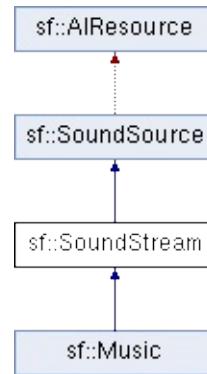
sf::SoundStream Class Reference abstract

[Audio module](#)

Abstract base class for streamed audio sources. More...

```
#include <SoundStream.hpp>
```

Inheritance diagram for sf::SoundStream:



Classes

struct `Chunk`

Structure defining a chunk of audio data to stream. [More...](#)

Public Types

enum Status { Stopped, Paused, Playing }

Enumeration of the sound source states. More...

Public Member Functions

virtual ~SoundStream ()
Destructor. [More...](#)

void play ()
Start or resume playing the audio stream. [More...](#)

void pause ()
Pause the audio stream. [More...](#)

void stop ()
Stop playing the audio stream. [More...](#)

unsigned int getChannelCount () const
Return the number of channels of the stream. [More...](#)

unsigned int getSampleRate () const
Get the stream sample rate of the stream. [More...](#)

Status getStatus () const
Get the current status of the stream (stopped, paused, play

void setPlayingOffset (Time timeOffset)
Change the current playing position of the stream. [More...](#)

Time getPlayingOffset () const
Get the current playing position of the stream. [More...](#)

void setLoop (bool loop)
Set whether or not the stream should loop after reaching th

bool getLoop () const

Tell whether or not the stream is in loop mode. More...

void `setPitch (float pitch)`
Set the pitch of the sound. More...

void `setVolume (float volume)`
Set the volume of the sound. More...

void `setPosition (float x, float y, float z)`
Set the 3D position of the sound in the audio scene. More..

void `setPosition (const Vector3f &position)`
Set the 3D position of the sound in the audio scene. More..

void `setRelativeToListener (bool relative)`
Make the sound's position relative to the listener or absolut

void `setMinDistance (float distance)`
Set the minimum distance of the sound. More...

void `setAttenuation (float attenuation)`
Set the attenuation factor of the sound. More...

float `getPitch () const`
Get the pitch of the sound. More...

float `getVolume () const`
Get the volume of the sound. More...

Vector3f `getPosition () const`
Get the 3D position of the sound in the audio scene. More..

bool `isRelativeToListener () const`
Tell whether the sound's position is relative to the listener o

```
float getMinDistance () const
```

Get the minimum distance of the sound. More...

```
float getAttenuation () const
```

Get the attenuation factor of the sound. More...

Protected Member Functions

`SoundStream ()`

Default constructor. [More...](#)

`void initialize (unsigned int channelCount, unsigned int sampleRate)`

Define the audio stream parameters. [More...](#)

`virtual bool onGetData (Chunk &data)=0`

Request a new chunk of audio samples from the stream source.

`virtual void onSeek (Time timeOffset)=0`

Change the current playing position in the stream source. [More...](#)

Protected Attributes

unsigned int `m_source`

OpenAL source identifier. [More...](#)

Detailed Description

Abstract base class for streamed audio sources.

Unlike audio buffers (see [sf::SoundBuffer](#)), audio streams are never comp

Instead, the audio data is acquired continuously while the stream is playi sound with no loading delay, and keeps the memory consumption very lov

Sound sources that need to be streamed are usually big files (compress hundreds of MB in memory) or files that would take a lot of time to be i network).

[sf::SoundStream](#) is a base class that doesn't care about the stream sc class. SFML provides a built-in specialization for big files (see [sf::Mus provided, but you can write your own by combining this class with the netv](#)

A derived class has to override two virtual functions:

- `onGetData` fills a new chunk of audio data to be played
- `onSeek` changes the current playing position in the source

It is important to note that each [SoundStream](#) is played in its own sepa loop doesn't block the rest of the program. In particular, the `OnGetData` & sometimes be called from this separate thread. It is important to keep this to take care of synchronization issues if you share data between threads.

Usage example:

```
class CustomStream : public sf::SoundStream
{
public:
```

```
bool open(const std::string& location)
{
// Open the source and get audio settings
    ...
unsigned int channelCount = ...;
unsigned int sampleRate = ...;

// Initialize the stream -- important!
initialize(channelCount, sampleRate);
}

private:

virtual bool onGetData(Chunk& data)
{
// Fill the chunk with audio data from the stream source
// (note: must not be empty if you want to continue playing)
    data.samples = ...;
    data.sampleCount = ...;

// Return true to continue playing
return true;
}

virtual void onSeek(Uint32 timeOffset)
{
// Change the current position in the stream source
    ...
}

// Usage
CustomStream stream;
stream.open("path/to/stream");
stream.play();
```

See also

[sf::Music](#)

Definition at line 45 of file [SoundStream.hpp](#).

Member Enumeration Documentation

enum sf::SoundSource::Status

Enumeration of the sound source states.

Enumerator	
Stopped	Sound is not playing.
Paused	Sound is paused.
Playing	Sound is playing.

Definition at line 50 of file SoundSource.hpp.

Constructor & Destructor Documentation

virtual sf::SoundStream::~SoundStream()

Destructor.

sf::SoundStream::SoundStream()

Default constructor.

This constructor is only meant to be called by derived classes.

Member Function Documentation

float sf::SoundSource::getAttenuation() const

Get the attenuation factor of the sound.

Returns

Attenuation factor of the sound

See also

[setAttenuation](#), [getMinDistance](#)

unsigned int sf::SoundStream::getChannelCount() const

Return the number of channels of the stream.

1 channel means a mono sound, 2 means stereo, etc.

Returns

Number of channels

bool sf::SoundStream::getLoop() const

Tell whether or not the stream is in loop mode.

Returns

True if the stream is looping, false otherwise

See also

[setLoop](#)

float sf::SoundSource::getMinDistance () const

Get the minimum distance of the sound.

Returns

Minimum distance of the sound

See also

[setMinDistance](#), [getAttenuation](#)

float sf::SoundSource::getPitch () const

Get the pitch of the sound.

Returns

Pitch of the sound

See also

[setPitch](#)

Time sf::SoundStream::getPlayingOffset () const

Get the current playing position of the stream.

Returns

Current playing position, from the beginning of the stream

See also

[setPlayingOffset](#)

Vector3f sf::SoundSource::getPosition () const

Get the 3D position of the sound in the audio scene.

Returns

Position of the sound

See also

[setPosition](#)

unsigned int sf::SoundStream::getSampleRate () const

Get the stream sample rate of the stream.

The sample rate is the number of audio samples played per second. The

Returns

Sample rate, in number of samples per second

Status sf::SoundStream::getStatus () const

Get the current status of the stream (stopped, paused, playing)

Returns

Current status

float sf::SoundSource::getVolume () const

Get the volume of the sound.

Returns

Volume of the sound, in the range [0, 100]

See also

[setVolume](#)

void sf::SoundStream::initialize (unsigned int channelCount, unsigned int sampleRate)

Define the audio stream parameters.

This function must be called by derived classes as soon as they know the play. Any attempt to manipulate the stream ([play\(\)](#), ...) before calling this multiple times if the settings of the audio stream change, but only when t

Parameters

channelCount Number of channels of the stream

sampleRate Sample rate, in samples per second

bool sf::SoundSource::isRelativeToListener () const

Tell whether the sound's position is relative to the listener or is absolute.

Returns

True if the position is relative, false if it's absolute

See also

[setRelativeToListener](#)

virtual bool sf::SoundStream::onGetData (Chunk & data)

Request a new chunk of audio samples from the stream source.

This function must be overridden by derived classes to provide the audio continuously by the streaming loop, in a separate thread. The source can stop the loop at any time, by returning false to the caller. If you return true (i.e. continue), make sure that the returned array of samples is not empty; this would stop the stream.

Parameters

data [Chunk](#) of data to fill

Returns

True to continue playback, false to stop

Implemented in [sf::Music](#).

virtual void sf::SoundStream::onSeek (Time timeOffset)

Change the current playing position in the stream source.

This function must be overridden by derived classes to allow random seeking.

Parameters

timeOffset New playing position, relative to the beginning of the stream.

Implemented in sf::Music.

void sf::SoundStream::pause ()

Pause the audio stream.

This function pauses the stream if it was playing, otherwise (stream already stopped), it does nothing. It has no effect.

See also

[play](#), [stop](#)

void sf::SoundStream::play ()

Start or resume playing the audio stream.

This function starts the stream if it was stopped, resumes it if it was paused, or continues it from the beginning if it was already playing. This function uses its own thread so it can be used while the program is played.

See also

[pause](#), [stop](#)

void sf::SoundSource::setAttenuation (float attenuation)

Set the attenuation factor of the sound.

The attenuation is a multiplicative factor which makes the sound more or less audible depending on the distance from the listener. An attenuation of 0 will produce a non-attenuated sound, while an attenuation of 1000 will produce a sound that is 1000 times less audible than the original.

always be the same whether it is heard from near or from far. On the contrary, a value such as 100 will make the sound fade out very quickly as it gets further away. The minimum distance is 0.0f and the maximum distance is infinity. The default value of the attenuation is 1.

Parameters

attenuation New attenuation factor of the sound

See also

[getAttenuation](#), [setMinDistance](#)

void sf::SoundStream::setLoop (bool loop)

Set whether or not the stream should loop after reaching the end.

If set, the stream will restart from beginning after reaching the end as if `setLoop(true)` was called. If not set, the stream will stop as if `setLoop(false)` was called. The default looping state for streams is false.

Parameters

loop True to play in loop, false to play once

See also

[getLoop](#)

void sf::SoundSource::setMinDistance (float distance)

Set the minimum distance of the sound.

The "minimum distance" of a sound is the maximum distance at which it can still be heard. Further than the minimum distance, it will start to fade out according to its attenuation. A negative value ("inside the head of the listener") is an invalid value and is forbidden. The minimum distance is 0.0f and the maximum distance is infinity. The default value of the minimum distance is 1.

Parameters

distance New minimum distance of the sound

See also

[getMinDistance](#), [setAttenuation](#)

void sf::SoundSource::setPitch (float pitch)

Set the pitch of the sound.

The pitch represents the perceived fundamental frequency of a sound; it acute or grave by changing its pitch. A side effect of changing the pitch the sound as well. The default value for the pitch is 1.

Parameters

pitch New pitch to apply to the sound

See also

[getPitch](#)

void sf::SoundStream::setPlayingOffset (Time timeOffset)

Change the current playing position of the stream.

The playing position can be changed when the stream is either paused position when the stream is stopped has no effect, since playing the stre

Parameters

timeOffset New playing position, from the beginning of the stream

See also

[getPlayingOffset](#)

```
void sf::SoundSource::setPosition ( float x,  
                                  float y,  
                                  float z  
    )
```

Set the 3D position of the sound in the audio scene.

Only sounds with one channel (mono sounds) can be spatialized. The default value is (0, 0, 0).

Parameters

- x** X coordinate of the position of the sound in the scene
- y** Y coordinate of the position of the sound in the scene
- z** Z coordinate of the position of the sound in the scene

See also

[getPosition](#)

```
void sf::SoundSource::setPosition ( const Vector3f & position )
```

Set the 3D position of the sound in the audio scene.

Only sounds with one channel (mono sounds) can be spatialized. The default value is (0, 0, 0).

Parameters

- position** Position of the sound in the scene

See also

[getPosition](#)

void sf::SoundSource::setRelativeToListener (bool relative)

Make the sound's position relative to the listener or absolute.

Making a sound relative to the listener will ensure that it will always be based on the position of the listener. This can be useful for non-spatialized sounds, sounds attached to the listener, or sounds attached to it. The default value is false (position is absolute).

Parameters

relative True to set the position relative, false to set it absolute

See also

[isRelativeToListener](#)

void sf::SoundSource::setVolume (float volume)

Set the volume of the sound.

The volume is a value between 0 (mute) and 100 (full volume). The default value is 100.

Parameters

volume Volume of the sound

See also

[getVolume](#)

void sf::SoundStream::stop ()

Stop playing the audio stream.

This function stops the stream if it was playing or paused, and does not also resets the playing position (unlike [pause\(\)](#)).

See also

[play](#), [pause](#)

Member Data Documentation

unsigned int sf::SoundSource::m_source

OpenAL source identifier.

Definition at line 274 of file [SoundSource.hpp](#).

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

- [SoundStream.hpp](#)
-

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::SoundStream::Chunk Struct Reference

Structure defining a chunk of audio data to stream. [More...](#)

```
#include <SoundStream.hpp>
```

Public Attributes

`const Int16 * samples`

Pointer to the audio samples. [More...](#)

`std::size_t sampleCount`

Number of samples pointed by Samples. [More...](#)

Detailed Description

Structure defining a chunk of audio data to stream.

Definition at line 53 of file [SoundStream.hpp](#).

Member Data Documentation

std::size_t sf::SoundStream::Chunk::sampleCount

Number of samples pointed by Samples.

Definition at line 56 of file [SoundStream.hpp](#).

const Int16* sf::SoundStream::Chunk::samples

Pointer to the audio samples.

Definition at line 55 of file [SoundStream.hpp](#).

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

- [SoundStream.hpp](#)
-

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

[Namespaces](#) | [Classes](#) | [Enumerations](#)

Graphics module

2D graphics module: sprites, text, shapes, ... [More...](#)

Namespaces

`sf::Gsl`

Namespace with GLSL types.

Classes

class [sf::BlendMode](#)

Blending modes for drawing. [More...](#)

class [sf::CircleShape](#)

Specialized shape representing a circle. [More...](#)

class [sf::Color](#)

Utility class for manipulating RGBA colors. [More...](#)

class [sf::ConvexShape](#)

Specialized shape representing a convex polygon. [More...](#)

class [sf::Drawable](#)

Abstract base class for objects that can be drawn to a render target. [More...](#)

class [sf::Font](#)

Class for loading and manipulating character fonts. [More...](#)

class [sf::Glyph](#)

Structure describing a glyph. [More...](#)

class [sf::Image](#)

Class for loading, manipulating and saving images. [More...](#)

class [sf::Rect< T >](#)

Utility class for manipulating 2D axis aligned rectangles. [More...](#)

class [sf::RectangleShape](#)

Specialized shape representing a rectangle. [More...](#)

class [sf::RenderStates](#)

Define the states used for drawing to a [RenderTarget](#). More...

class [sf::RenderTarget](#)

Base class for all render targets (window, texture, ...) More...

class [sf::RenderTexture](#)

Target for off-screen 2D rendering into a texture. More...

class [sf::RenderWindow](#)

Window that can serve as a target for 2D drawing. More...

class [sf::Shader](#)

Shader class (vertex, geometry and fragment) More...

class [sf::Shape](#)

Base class for textured shapes with outline. More...

class [sf::Sprite](#)

Drawable representation of a texture, with its own transformations

class [sf::Text](#)

Graphical text that can be drawn to a render target. More...

class [sf::Texture](#)

Image living on the graphics card that can be used for drawing. More...

class [sf::Transform](#)

Define a 3x3 transform matrix. More...

class [sf::Transformable](#)

Decomposed transform defined by a position, a rotation and a scale.

class [sf::Vertex](#)

Define a point with color and texture coordinates. More...

class [sf::VertexArray](#)

Define a set of one or more 2D primitives. [More...](#)

class [sf::View](#)

2D camera that defines what region is shown on screen [More...](#)

Enumerations

```
sf::PrimitiveType {  
    sf::Points, sf::Lines, sf::LineStrip, sf::Triangles,  
enum    sf::TriangleStrip, sf::TriangleFan, sf::Quads, sf::LinesStrip = Lin  
        sf::TrianglesStrip = TriangleStrip, sf::TrianglesFan = TriangleFa  
    }  
Types of primitives that a sf::VertexArray can render. More...
```

Detailed Description

2D graphics module: sprites, text, shapes, ...

Enumeration Type Documentation

enum sf::PrimitiveType

Types of primitives that a `sf::VertexArray` can render.

Points and lines have no area, therefore their thickness will always be transform and view.

Enumerator

Points	List of individual points.
Lines	List of individual lines.
LineStrip	List of connected lines, a point uses the previous point
Triangles	List of individual triangles.
TriangleStrip	List of connected triangles, a point uses the two previous points
TriangleFan	List of connected triangles, a point uses the common previous point to form a triangle.
Quads	List of individual quads (deprecated, don't work with OpenGL)

LinesStrip

Deprecated:
Use LineStrip instead

TrianglesStrip

Deprecated:
Use TriangleStrip instead

TrianglesFan

Deprecated:
Use TriangleFan instead

Definition at line 39 of file [PrimitiveType.hpp](#).

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

Namespaces

[Classes](#)

[Files](#)

[Namespace List](#)

[Namespace Members](#)

[Typedefs](#)

sf::Gsl Namespace Reference

[Graphics module](#)

Namespace with GLSL types. [More...](#)

Typedefs

typedef `Vector2< float >` **Vec2**
2D float vector (`vec2` in GLSL) [More...](#)

typedef `Vector2< int >` **Ivec2**
2D int vector (`ivec2` in GLSL) [More...](#)

typedef `Vector2< bool >` **Bvec2**
2D bool vector (`bvec2` in GLSL) [More...](#)

typedef `Vector3< float >` **Vec3**
3D float vector (`vec3` in GLSL) [More...](#)

typedef `Vector3< int >` **Ivec3**
3D int vector (`ivec3` in GLSL) [More...](#)

typedef `Vector3< bool >` **Bvec3**
3D bool vector (`bvec3` in GLSL) [More...](#)

typedef implementation defined **Vec4**
4D float vector (`vec4` in GLSL) [More...](#)

typedef implementation defined **Ivec4**
4D int vector (`ivec4` in GLSL) [More...](#)

typedef implementation defined **Bvec4**
4D bool vector (`bvec4` in GLSL) [More...](#)

typedef implementation defined **Mat3**
3x3 float matrix (`mat3` in GLSL) [More...](#)

typedef implementation defined **Mat4**

4x4 float matrix (`mat4` in GLSL) [More...](#)

Detailed Description

Namespace with GLSL types.

The `sf::Gsl` namespace contains types that match their equivalents language. These types are exclusively used by the `sf::Shader` class.

Types that already exist in SFML, such as `sf::Vector2<T>` and `sf::Vector`, you can use the types in this namespace as well as the original ones. C `Gsl::Vec4` or `Gsl::Mat3`. Their actual type is an implementation detail and

All vector types support a default constructor that initializes every component constructor with one parameter for each component. The components are called x, y, z, and w.

All matrix types support a constructor with a `float*` parameter that points to size (that is, 9 in a 3x3 matrix, 16 in a 4x4 matrix). Furthermore, they can hold objects.

See also

`sf::Shader`

Typedef Documentation

typedef Vector2<bool> sf::Gsl::Bvec2

2D bool vector (bvec2 in GLSL)

Definition at line 76 of file Gsl.hpp.

typedef Vector3<bool> sf::Gsl::Bvec3

3D bool vector (bvec3 in GLSL)

Definition at line 94 of file Gsl.hpp.

typedef implementation defined sf::Gsl::Bvec4

4D bool vector (bvec4 in GLSL)

Definition at line 130 of file Gsl.hpp.

typedef Vector2<int> sf::Gsl::Ivec2

2D int vector (ivec2 in GLSL)

Definition at line 70 of file `Glsl.hpp`.

typedef Vector3<int> sf::Glsl::ivec3

3D int vector (`ivec3` in GLSL)

Definition at line 88 of file `Glsl.hpp`.

typedef implementation defined sf::Glsl::ivec4

4D int vector (`ivec4` in GLSL)

4D int vectors can be implicitly converted from `sf::Color` instances unchanged inside the integer interval [0, 255].

```
sf::Glsl::ivec4 zeroVector;
sf::Glsl::ivec4 vector(1, 2, 3, 4);
sf::Glsl::ivec4 color = sf::Color::Cyan;
```

Definition at line 124 of file `Glsl.hpp`.

typedef implementation defined sf::Glsl::Mat3

3x3 float matrix (`mat3` in GLSL)

The matrix can be constructed from an array with 3x3 elements, alig example, a translation by (x, y) looks as follows:

```
float array[9] =
{
    1, 0, 0,
```

```
    0,  1,  0,
    x,  y,  1
};

sf::Gsls::Mat3 matrix(array);
```

Mat3 can also be implicitly converted from sf::Transform:

```
sf::Transform transform;
sf::Gsls::Mat3 matrix = transform;
```

Definition at line 155 of file Gsls.hpp.

typedef implementation defined sf::Gsls::Mat4

4x4 float matrix (mat4 in GLSL)

The matrix can be constructed from an array with 4x4 elements, alig example, a translation by (x, y, z) looks as follows:

```
float array[16] =
{
    1,  0,  0,  0,
    0,  1,  0,  0,
    0,  0,  1,  0,
    x,  y,  z,  1
};

sf::Gsls::Mat4 matrix(array);
```

Mat4 can also be implicitly converted from sf::Transform:

```
sf::Transform transform;
sf::Gsls::Mat4 matrix = transform;
```

Definition at line 181 of file Gsls.hpp.

typedef Vector2<float> sf::Gsl::Vec2

2D float vector (vec2 in GLSL)

Definition at line 64 of file Gsl.hpp.

typedef Vector3<float> sf::Gsl::Vec3

3D float vector (vec3 in GLSL)

Definition at line 82 of file Gsl.hpp.

typedef implementation defined sf::Gsl::Vec4

4D float vector (vec4 in GLSL)

4D float vectors can be implicitly converted from sf::Color instances. E from integers in [0, 255] to floating point values in [0, 1].

```
sf::Gsl::Vec4 zeroVector;
sf::Gsl::Vec4 vector(1.f, 2.f, 3.f, 4.f);
sf::Gsl::Vec4 color = sf::Color::Cyan;
```

Definition at line 110 of file Gsl.hpp.

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [Public Attributes](#) | [Related Functions](#) | [List of all members](#)

sf::BlendMode Class Reference

[Graphics module](#)

Blending modes for drawing. More...

```
#include <BlendMode.hpp>
```

Public Types

```
enum Factor {  
    Zero, One, SrcColor, OneMinusSrcColor,  
    DstColor, OneMinusDstColor, SrcAlpha, OneMinusSrcAlpha,  
    DstAlpha, OneMinusDstAlpha  
}
```

Enumeration of the blending factors. [More...](#)

```
enum Equation { Add, Subtract, ReverseSubtract }
```

Enumeration of the blending equations. [More...](#)

Public Member Functions

`BlendMode ()`

Default constructor. More...

`BlendMode (Factor sourceFactor, Factor destinationFactor, Equation`

Construct the blend mode given the factors and equation. More...

`BlendMode (Factor colorSourceFactor, Factor colorDestinationFactor,`

`Factor alphaSourceFactor, Factor alphaDestinationFactor, Equation a`

Construct the blend mode given the factors and equation. More...

Public Attributes

Factor **colorSrcFactor**

Source blending factor for the color channels. [More...](#)

Factor **colorDstFactor**

Destination blending factor for the color channels. [More...](#)

Equation **colorEquation**

Blending equation for the color channels. [More...](#)

Factor **alphaSrcFactor**

Source blending factor for the alpha channel. [More...](#)

Factor **alphaDstFactor**

Destination blending factor for the alpha channel. [More...](#)

Equation **alphaEquation**

Blending equation for the alpha channel. [More...](#)

Related Functions

(Note that these are not member functions.)

`bool operator==(const BlendMode &left, const BlendMode &right)`
Overload of the == operator. More...

`bool operator!=(const BlendMode &left, const BlendMode &right)`
Overload of the != operator. More...

Detailed Description

Blending modes for drawing.

`sf::BlendMode` is a class that represents a blend mode.

A blend mode determines how the colors of an object you draw are mixed with the buffer.

The class is composed of 6 components, each of which has its own public

- Color Source Factor (`colorSrcFactor`)
- Color Destination Factor (`colorDstFactor`)
- Color Blend Equation (`colorEquation`)
- Alpha Source Factor (`alphaSrcFactor`)
- Alpha Destination Factor (`alphaDstFactor`)
- Alpha Blend Equation (`alphaEquation`)

The source factor specifies how the pixel you are drawing contributes to the resulting pixel. The destination factor specifies how the pixel already drawn in the buffer contributes to the resulting pixel.

The color channels RGB (red, green, blue; simply referred to as color) and alpha channel (alpha) can be treated separately. This separation can be useful for specific blend modes that need it and will simply treat the color as a single unit.

The blend factors and equations correspond to their OpenGL equivalents. The resulting pixel is calculated according to the following formula (`src` is the color of the destination pixel, the other variables correspond to the public members being + or - operators):

```
dst.rgb = colorSrcFactor * src.rgb (colorEquation) colorDstFactor * ds  
dst.a   = alphaSrcFactor * src.a   (alphaEquation) alphaDstFactor * ds
```

All factors and colors are represented as floating point numbers between 0.0f and 1.0f. The result is clamped to fit in that range.

The most common blending modes are defined as constants in the sf::BlendMode enum:

```
sf::BlendMode alphaBlending      = sf::BlendAlpha;  
sf::BlendMode additiveBlending  = sf::BlendAdd;  
sf::BlendMode multiplicativeBlending = sf::BlendMultiply;  
sf::BlendMode noBlending        = sf::BlendNone;
```

In SFML, a blend mode can be specified every time you draw a [sf::Drawable](#) object by setting it as the `blendMode` part of the [sf::RenderStates](#) compound that is passed to the member functions of the [sf::RenderTarget](#).

See also

[sf::RenderStates](#), [sf::RenderTarget](#)

Definition at line 41 of file [BlendMode.hpp](#).

Member Enumeration Documentation

enum sf::BlendMode::Equation

Enumeration of the blending equations.

The equations are mapped directly to their OpenGL equivalents, see `glBlendEquationSeparate()`.

Enumerator

Add	$\text{Pixel} = \text{Src} * \text{SrcFactor} + \text{Dst} * \text{DstFactor}.$
Subtract	$\text{Pixel} = \text{Src} * \text{SrcFactor} - \text{Dst} * \text{DstFactor}.$
ReverseSubtract	$\text{Pixel} = \text{Dst} * \text{DstFactor} - \text{Src} * \text{SrcFactor}.$

Definition at line 69 of file `BlendMode.hpp`.

enum sf::BlendMode::Factor

Enumeration of the blending factors.

The factors are mapped directly to their OpenGL equivalents, see `glBlendFuncSeparate()`.

Enumerator	
Zero	(0, 0, 0, 0)
One	(1, 1, 1, 1)
SrcColor	(src.r, src.g, src.b, src.a)
OneMinusSrcColor	(1, 1, 1, 1) - (src.r, src.g, src.b, src.a)
DstColor	(dst.r, dst.g, dst.b, dst.a)
OneMinusDstColor	(1, 1, 1, 1) - (dst.r, dst.g, dst.b, dst.a)
SrcAlpha	(src.a, src.a, src.a, src.a)
OneMinusSrcAlpha	(1, 1, 1, 1) - (src.a, src.a, src.a, src.a)
DstAlpha	(dst.a, dst.a, dst.a, dst.a)
OneMinusDstAlpha	(1, 1, 1, 1) - (dst.a, dst.a, dst.a, dst.a)

Definition at line 49 of file [BlendMode.hpp](#).

Constructor & Destructor Documentation

sf::BlendMode::BlendMode ()

Default constructor.

Constructs a blending mode that does alpha blending.

Construct the blend mode given the factors and equation.

This constructor uses the same factors and equation for both color defaults to the Add equation.

Parameters

sourceFactor	Specifies how to compute the source factor for the channels.
destinationFactor	Specifies how to compute the destination factor channels.
blendEquation	Specifies how to combine the source and destination factors.

**sf::BlendMode::BlendMode (Factor colorSourceFactor,
Factor colorDestinationFactor,**

```
Equation colorBlendEquation,  
Factor alphaSourceFactor,  
Factor alphaDestinationFactor,  
Equation alphaBlendEquation  
)
```

Construct the blend mode given the factors and equation.

Parameters

colorSourceFactor	Specifies how to compute the source factor.
colorDestinationFactor	Specifies how to compute the destination factor.
colorBlendEquation	Specifies how to combine the source and destination colors.
alphaSourceFactor	Specifies how to compute the source factor.
alphaDestinationFactor	Specifies how to compute the destination factor.
alphaBlendEquation	Specifies how to combine the source and destination alphas.

Friends And Related Function Documentation

```
bool operator!= ( const BlendMode & left,  
                  const BlendMode & right  
                )
```

Overload of the != operator.

Parameters

left Left operand
right Right operand

Returns

True if blending modes are different, false if they are equal

```
bool operator==( const BlendMode & left,  
                   const BlendMode & right  
                 )
```

Overload of the == operator.

Parameters

left Left operand
right Right operand

Returns

True if blending modes are equal, false if they are different

Member Data Documentation

Factor sf::BlendMode::alphaDstFactor

Destination blending factor for the alpha channel.

Definition at line [119](#) of file BlendMode.hpp.

Equation sf::BlendMode::alphaEquation

Blending equation for the alpha channel.

Definition at line [120](#) of file BlendMode.hpp.

Factor sf::BlendMode::alphaSrcFactor

Source blending factor for the alpha channel.

Definition at line [118](#) of file BlendMode.hpp.

Factor sf::BlendMode::colorDstFactor

Destination blending factor for the color channels.

Definition at line [116](#) of file BlendMode.hpp.

Equation sf::BlendMode::colorEquation

Blending equation for the color channels.

Definition at line [117](#) of file `BlendMode.hpp`.

Factor sf::BlendMode::colorSrcFactor

Source blending factor for the color channels.

Definition at line [115](#) of file `BlendMode.hpp`.

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

- `BlendMode.hpp`
-

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Protected Member Functions](#) | [List of all members](#)

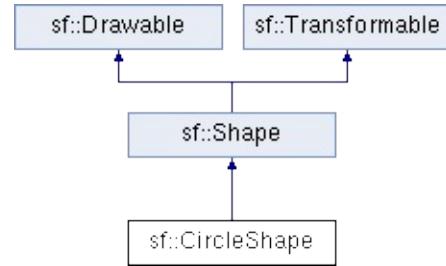
sf::CircleShape Class Reference

Graphics module

Specialized shape representing a circle. [More...](#)

```
#include <CircleShape.hpp>
```

Inheritance diagram for sf::CircleShape:



Public Member Functions

`CircleShape (float radius=0, std::size_t pointCount=32)`
Default constructor. [More...](#)

`void setRadius (float radius)`
Set the radius of the circle. [More...](#)

`float getRadius () const`
Get the radius of the circle. [More...](#)

`void setPointCount (std::size_t count)`
Set the number of points of the circle. [More...](#)

`virtual std::size_t getPointCount () const`
Get the number of points of the circle. [More...](#)

`virtual Vector2f getPoint (std::size_t index) const`
Get a point of the circle. [More...](#)

`void setTexture (const Texture *texture, bool resetRect=false)`
Change the source texture of the shape. [More...](#)

`void setTextureRect (const IntRect &rect)`
Set the sub-rectangle of the texture that the shape will use.

`void setFillColor (const Color &color)`
Set the fill color of the shape. [More...](#)

`void setOutlineColor (const Color &color)`
Set the outline color of the shape. [More...](#)

`void setOutlineThickness (float thickness)`

Set the thickness of the shape's outline. More...

const Texture * **getTexture () const**

Get the source texture of the shape. More...

const IntRect & **getTextureRect () const**

Get the sub-rectangle of the texture displayed by the

const Color & **getFillColor () const**

Get the fill color of the shape. More...

const Color & **getOutlineColor () const**

Get the outline color of the shape. More...

float **getOutlineThickness () const**

Get the outline thickness of the shape. More...

FloatRect **getLocalBounds () const**

Get the local bounding rectangle of the entity. More..

FloatRect **getGlobalBounds () const**

Get the global (non-minimal) bounding rectangle of the

void **setPosition (float x, float y)**

set the position of the object More...

void **setPosition (const Vector2f &position)**

set the position of the object More...

void **setRotation (float angle)**

set the orientation of the object More...

void **setScale (float factorX, float factorY)**

set the scale factors of the object More...

`void setScale (const Vector2f &factors)`
set the scale factors of the object More...

`void setOrigin (float x, float y)`
set the local origin of the object More...

`void setOrigin (const Vector2f &origin)`
set the local origin of the object More...

`const Vector2f & getPosition () const`
get the position of the object More...

`float getRotation () const`
get the orientation of the object More...

`const Vector2f & getScale () const`
get the current scale of the object More...

`const Vector2f & getOrigin () const`
get the local origin of the object More...

`void move (float offsetX, float offsetY)`
Move the object by a given offset. More...

`void move (const Vector2f &offset)`
Move the object by a given offset. More...

`void rotate (float angle)`
Rotate the object. More...

`void scale (float factorX, float factorY)`
Scale the object. More...

`void scale (const Vector2f &factor)`
Scale the object. More...

```
const Transform & getTransform () const  
    get the combined transform of the object More...
```

```
const Transform & getInverseTransform () const  
    get the inverse of the combined transform of the obje
```

Protected Member Functions

void `update ()`

Recompute the internal geometry of the shape. [More...](#)

Detailed Description

Specialized shape representing a circle.

This class inherits all the functions of `sf::Transformable` (position, rotation functions of `sf::Shape` (outline, color, texture, ...)).

Usage example:

```
sf::CircleShape circle;
circle.setRadius(150);
circle.setOutlineColor(sf::Color::Red);
circle.setOutlineThickness(5);
circle.setPosition(10, 20);
...
window.draw(circle);
```

Since the graphics card can't draw perfect circles, we have to fake them to each other. The "points count" property of `sf::CircleShape` defines how and therefore defines the quality of the circle.

The number of points can also be used for another purpose; with small number of points it creates a polygon shape: equilateral triangle, square, pentagon, hexagon, ...

See also

`sf::Shape`, `sf::RectangleShape`, `sf::ConvexShape`

Definition at line 41 of file `CircleShape.hpp`.

Constructor & Destructor Documentation

```
sf::CircleShape::CircleShape ( float      radius = 0,  
                           std::size_t pointCount = 30  
                         )
```

Default constructor.

Parameters

radius Radius of the circle
pointCount Number of points composing the circle

Member Function Documentation

`const Color& sf::Shape::getFillColor() const`

Get the fill color of the shape.

Returns

Fill color of the shape

See also

[setFillColor](#)

`FloatRect sf::Shape::getGlobalBounds() const`

Get the global (non-minimal) bounding rectangle of the entity.

The returned rectangle is in global coordinates, which means it returns the bounds of the shape in the global 2D world's coordinate system. This function does not necessarily return the *minimal* bounding rectangle covers all the vertices (but possibly more). This allows bounds as a first check; you may want to use more precise checks on top of this.

Returns

Global bounding rectangle of the entity

const Transform& sf::Transformable::getInverseTransform() const

get the inverse of the combined transform of the object

Returns

Inverse of the combined transformations applied to the object

See also

[getTransform](#)

FloatRect sf::Shape::getLocalBounds() const

Get the local bounding rectangle of the entity.

The returned rectangle is in local coordinates, which means that (translation, rotation, scale, ...) that are applied to the entity. In other bounds of the entity in the entity's coordinate system.

Returns

Local bounding rectangle of the entity

const Vector2f& sf::Transformable::getOrigin() const

get the local origin of the object

Returns

Current origin

See also

[setOrigin](#)

const Color& sf::Shape::getOutlineColor() const

Get the outline color of the shape.

Returns

Outline color of the shape

See also

[setOutlineColor](#)

float sf::Shape::getOutlineThickness() const

Get the outline thickness of the shape.

Returns

Outline thickness of the shape

See also

[setOutlineThickness](#)

virtual Vector2f sf::CircleShape::getPoint(std::size_t index) const

Get a point of the circle.

The returned point is in local coordinates, that is, the shape's transforms taken into account. The result is undefined if *index* is out of the valid range [0 .. `getPointCount()` - 1].

Parameters

index Index of the point to get, in range [0 .. `getPointCount()` - 1]

Returns

index-th point of the shape

Implements [sf::Shape](#).

virtual std::size_t sf::CircleShape::getPointCount() const

Get the number of points of the circle.

Returns

Number of points of the circle

See also

[setPointCount](#)

Implements [sf::Shape](#).

const Vector2f& sf::Transformable::getPosition() const

get the position of the object

Returns

Current position

See also

[setPosition](#)

float sf::CircleShape::getRadius() const

Get the radius of the circle.

Returns

Radius of the circle

See also

[setRadius](#)

float sf::Transformable::getRotation() const

get the orientation of the object

The rotation is always in the range [0, 360].

Returns

Current rotation, in degrees

See also

[setRotation](#)

const Vector2f& sf::Transformable::getScale() const

get the current scale of the object

Returns

Current scale factors

See also

[setScale](#)

const Texture* sf::Shape::getTexture() const

Get the source texture of the shape.

If the shape has no source texture, a NULL pointer is returned. This means that you can't modify the texture when you retrieve it with this function.

Returns

Pointer to the shape's texture

See also

[setTexture](#)

const IntRect& sf::Shape::getTextureRect() const

Get the sub-rectangle of the texture displayed by the shape.

Returns

Texture rectangle of the shape

See also

[setTextureRect](#)

const Transform& sf::Transformable::getTransform() const

get the combined transform of the object

Returns

Transform combining the position/rotation/scale/origin of the object

See also

[getInverseTransform](#)

```
void sf::Transformable::move ( float offsetX,
                           float offsetY
)
```

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition` equivalent to the following code:

```
sf::Vector2f pos = object.getPosition();
object.setPosition(pos.x + offsetX, pos.y + offsetY);
```

Parameters

offsetX X offset
offsetY Y offset

See also

`setPosition`

```
void sf::Transformable::move ( const Vector2f & offset )
```

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition` equivalent to the following code:

```
object.setPosition(object.getPosition() + offset);
```

Parameters

offset Offset

See also

[setPosition](#)

void sf::Transformable::rotate (float angle)

Rotate the object.

This function adds to the current rotation of the object, unlike `setRotation` which is equivalent to the following code:

```
object.setRotation(object.getRotation() + angle);
```

Parameters

angle Angle of rotation, in degrees

void sf::Transformable::scale (float factorX, float factorY)

Scale the object.

This function multiplies the current scale of the object, unlike `setScale` which is equivalent to the following code:

```
sf::Vector2f scale = object.getScale();  
object.setScale(scale.x * factorX, scale.y * factorY);
```

Parameters

factorX Horizontal scale factor

factorY Vertical scale factor

See also

[setScale](#)

void sf::Transformable::scale (const Vector2f & factor)

Scale the object.

This function multiplies the current scale of the object, unlike `setScale`, it is equivalent to the following code:

```
sf::Vector2f scale = object.getScale();
object.setScale(scale.x * factor.x, scale.y * factor.y);
```

Parameters

factor Scale factors

See also

[setScale](#)

void sf::Shape::setFillColor (const Color & color)

Set the fill color of the shape.

This color is modulated (multiplied) with the shape's texture if any. It can also change its global opacity. You can use `sf::Color::Transparent` to make the shape transparent, and have the outline alone. By default, the shape's fill color is black.

Parameters

color New color of the shape

See also

`getFillColor`, `setOutlineColor`

```
void sf::Transformable::setOrigin ( float x,
                                    float y
                                )
```

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transfor

Parameters

- x** X coordinate of the new origin
- y** Y coordinate of the new origin

See also

`getOrigin`

```
void sf::Transformable::setOrigin ( const Vector2f & origin )
```

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transfor

Parameters

- origin** New origin

See also

getOrigin

```
void sf::Shape::setOutlineColor ( const Color & color )
```

Set the outline color of the shape.

By default, the shape's outline color is opaque white.

Parameters

color New outline color of the shape

See also

[getOutlineColor](#), [setFillColor](#)

```
void sf::Shape::setOutlineThickness ( float thickness )
```

Set the thickness of the shape's outline.

Note that negative values are allowed (so that the outline expands toward the center). Using zero disables the outline. By default, the outline thickness is 0.

Parameters

thickness New outline thickness

See also

[getOutlineThickness](#)

```
void sf::CircleShape::setPointCount ( std::size_t count )
```

Set the number of points of the circle.

Parameters

count New number of points of the circle

See also

[getPointCount](#)

```
void sf::Transformable::setPosition ( float x,
                                    float y
                                )
```

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

x X coordinate of the new position
y Y coordinate of the new position

See also

[move](#), [getPosition](#)

```
void sf::Transformable::setPosition ( const Vector2f & position )
```

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

position New position

See also

[move](#), [getPosition](#)

void sf::CircleShape::setRadius (float radius)

Set the radius of the circle.

Parameters

radius New radius of the circle

See also

[getRadius](#)

void sf::Transformable::setRotation (float angle)

set the orientation of the object

This function completely overwrites the previous rotation. See the [rotate](#) function on the previous rotation instead. The default rotation of a transformable object is 0 degrees.

Parameters

angle New rotation, in degrees

See also

[rotate](#), [getRotation](#)

```
void sf::Transformable::setScale ( float factorX,  
                                 float factorY  
                               )
```

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function for the previous scale instead. The default scale of a transformable object is

Parameters

factorX New horizontal scale factor

factory New vertical scale factor

See also

scale, getScale

```
void sf::Transformable::setScale ( const Vector2f & factors )
```

set the scale factors of the object

This function completely overwrites the previous scale. See the `scale` function for the previous scale instead. The default scale of a transformable object is

Parameters

factors New scale factors

See also

scale, getScale

```
void sf::Shape::setTexture ( const Texture * texture,  
                           bool                  resetRect = false )
```

)

Change the source texture of the shape.

The *texture* argument refers to a texture that must exist as long as the shape does. The shape doesn't store its own copy of the texture, but rather keeps a pointer to it. If the source texture is destroyed and the shape tries to use it, the texture can be NULL to disable texturing. If *resetRect* is true, the TextureRect of the shape is automatically adjusted to the size of the new texture. If it is false, the texture rect is not modified.

Parameters

texture New texture

resetRect Should the texture rect be reset to the size of the new texture?

See also

[getTexture](#), [setTextureRect](#)

void sf::Shape::setTextureRect(const IntRect & rect)

Set the sub-rectangle of the texture that the shape will display.

The texture rect is useful when you don't want to display the whole texture. By default, the texture rect covers the entire texture.

Parameters

rect Rectangle defining the region of the texture to display

See also

[getTextureRect](#), [setTexture](#)

void sf::Shape::update()

Recompute the internal geometry of the shape.

This function must be called by the derived class everytime the shape's either getPointCount or getPoint is different).

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

- [CircleShape.hpp](#)
-

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Member Functions](#) | [Public Attributes](#) | [Static Public Attributes](#) | [Related Functions](#) | [List of all members](#)

sf::Color Class Reference

Graphics module

Utility class for manipulating RGBA colors. More...

```
#include <Color.hpp>
```

Public Member Functions

[Color \(\)](#)

Default constructor. [More...](#)

[Color \(Uint8 red, Uint8 green, Uint8 blue, Uint8 alpha=255\)](#)

Construct the color from its 4 RGBA components. [More...](#)

[Color \(Uint32 color\)](#)

Construct the color from 32-bit unsigned integer. [More...](#)

[Uint32 toInteger \(\) const](#)

Retrieve the color as a 32-bit unsigned integer. [More...](#)

Public Attributes

Uint8 `r`

Red component. [More...](#)

Uint8 `g`

Green component. [More...](#)

Uint8 `b`

Blue component. [More...](#)

Uint8 `a`

Alpha (opacity) component. [More...](#)

Static Public Attributes

static const **Color** **Black**
Black predefined color. [More...](#)

static const **Color** **White**
White predefined color. [More...](#)

static const **Color** **Red**
Red predefined color. [More...](#)

static const **Color** **Green**
Green predefined color. [More...](#)

static const **Color** **Blue**
Blue predefined color. [More...](#)

static const **Color** **Yellow**
Yellow predefined color. [More...](#)

static const **Color** **Magenta**
Magenta predefined color. [More...](#)

static const **Color** **Cyan**
Cyan predefined color. [More...](#)

static const **Color** **Transparent**
Transparent (black) predefined color. [More...](#)

Related Functions

(Note that these are not member functions.)

`bool operator==(const Color &left, const Color &right)`
Overload of the == operator. [More...](#)

`bool operator!=(const Color &left, const Color &right)`
Overload of the != operator. [More...](#)

`Color operator+(const Color &left, const Color &right)`
Overload of the binary + operator. [More...](#)

`Color operator-(const Color &left, const Color &right)`
Overload of the binary - operator. [More...](#)

`Color operator*(const Color &left, const Color &right)`
Overload of the binary * operator. [More...](#)

`Color & operator+=(Color &left, const Color &right)`
Overload of the binary += operator. [More...](#)

`Color & operator-=(Color &left, const Color &right)`
Overload of the binary -= operator. [More...](#)

`Color & operator*=(Color &left, const Color &right)`
Overload of the binary *= operator. [More...](#)

Detailed Description

Utility class for manipulating RGBA colors.

`sf::Color` is a simple color class composed of 4 components:

- Red
- Green
- Blue
- Alpha (opacity)

Each component is a public member, an unsigned integer in the range [0, 255] constructed and manipulated very easily:

```
sf::Color color(255, 0, 0); // red
color.r = 0;                // make it black
color.b = 128;              // make it dark blue
```

The fourth component of colors, named "alpha", represents the opacity of the color. A value of 255 will be fully opaque, while an alpha value of 0 will make a color transparent. The value of the other components is unaffected.

The most common colors are already defined as static variables:

```
sf::Color black      = sf::Color::Black;
sf::Color white     = sf::Color::White;
sf::Color red       = sf::Color::Red;
sf::Color green    = sf::Color::Green;
sf::Color blue     = sf::Color::Blue;
sf::Color yellow   = sf::Color::Yellow;
sf::Color magenta  = sf::Color::Magenta;
sf::Color cyan     = sf::Color::Cyan;
sf::Color transparent = sf::Color::Transparent;
```

Colors can also be added and modulated (multiplied) using the overloaded operators.

Definition at line 40 of file [Color.hpp](#).

Constructor & Destructor Documentation

`sf::Color::Color()`

Default constructor.

Constructs an opaque black color. It is equivalent to `sf::Color(0, 0, 0, 255)`.

```
sf::Color::Color( Uint8 red,  
                  Uint8 green,  
                  Uint8 blue,  
                  Uint8 alpha = 255  
                )
```

Construct the color from its 4 RGBA components.

Parameters

- red** Red component (in the range [0, 255])
- green** Green component (in the range [0, 255])
- blue** Blue component (in the range [0, 255])
- alpha** Alpha (opacity) component (in the range [0, 255])

`sf::Color::Color(UInt32 color)`

Construct the color from 32-bit unsigned integer.

Parameters

color Number containing the RGBA components (in that order)

Member Function Documentation

Uint32 sf::Color::toInteger() const

Retrieve the color as a 32-bit unsigned integer.

Returns

Color represented as a 32-bit unsigned integer

Friends And Related Function Documentation

```
bool operator!= ( const Color & left,  
                  const Color & right  
                )
```

Overload of the != operator.

This operator compares two colors and check if they are different.

Parameters

left Left operand
right Right operand

Returns

True if colors are different, false if they are equal

```
Color operator* ( const Color & left,  
                  const Color & right  
                )
```

Overload of the binary * operator.

This operator returns the component-wise multiplication (also called Hadamard product). Components are then divided by 255 so that the result is still in the range [0, 1].

Parameters

left Left operand

right Right operand

Returns

Result of *left* * *right*

```
Color & operator*=( Color & left,  
                      const Color & right  
                    )
```

Overload of the binary *= operator.

This operator returns the component-wise multiplication (also called "assigns the result to the left operand. Components are then divided by 2 range [0, 255].

Parameters

left Left operand

right Right operand

Returns

Reference to *left*

```
Color operator+ ( const Color & left,  
                  const Color & right  
                )
```

Overload of the binary + operator.

This operator returns the component-wise sum of two colors. Componer to 255.

Parameters

left Left operand
right Right operand

Returns

Result of *left* + *right*

```
Color & operator+=( Color & left,  
                      const Color & right  
                    )
```

Overload of the binary += operator.

This operator computes the component-wise sum of two colors, and returns a new operand. Components that exceed 255 are clamped to 255.

Parameters

left Left operand
right Right operand

Returns

Reference to *left*

```
Color operator-( const Color & left,  
                  const Color & right  
                )
```

Overload of the binary - operator.

This operator returns the component-wise subtraction of two colors. Components that exceed 255 are clamped to 255.

Parameters

left Left operand
right Right operand

Returns

Result of *left* - *right*

```
Color & operator-= ( Color & left,  
                      const Color & right  
                    )
```

Overload of the binary -= operator.

This operator computes the component-wise subtraction of two colors, *left* - *right*. Components below 0 are clamped to 0.

Parameters

left Left operand
right Right operand

Returns

Reference to *left*

```
bool operator==( const Color & left,  
                   const Color & right  
                 )
```

Overload of the == operator.

This operator compares two colors and check if they are equal.

Parameters

left Left operand

right Right operand

Returns

True if colors are equal, false if they are different

Member Data Documentation

Uint8 sf::Color::a

Alpha (opacity) component.

Definition at line [99](#) of file [Color.hpp](#).

Uint8 sf::Color::b

Blue component.

Definition at line [98](#) of file [Color.hpp](#).

const Color sf::Color::Black

Black predefined color.

Definition at line [83](#) of file [Color.hpp](#).

const Color sf::Color::Blue

Blue predefined color.

Definition at line [87](#) of file [Color.hpp](#).

const Color sf::Color::Cyan

Cyan predefined color.

Definition at line [90](#) of file [Color.hpp](#).

Uint8 sf::Color::g

Green component.

Definition at line [97](#) of file [Color.hpp](#).

const Color sf::Color::Green

Green predefined color.

Definition at line [86](#) of file [Color.hpp](#).

const Color sf::Color::Magenta

Magenta predefined color.

Definition at line [89](#) of file [Color.hpp](#).

Uint8 sf::Color::r

Red component.

Definition at line [96](#) of file [Color.hpp](#).

const Color sf::Color::Red

Red predefined color.

Definition at line [85](#) of file [Color.hpp](#).

const Color sf::Color::Transparent

Transparent (black) predefined color.

Definition at line [91](#) of file [Color.hpp](#).

const Color sf::Color::White

White predefined color.

Definition at line [84](#) of file [Color.hpp](#).

const Color sf::Color::Yellow

Yellow predefined color.

Definition at line [88](#) of file [Color.hpp](#).

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

- [Color.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#).

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Protected Member Functions](#) | [List of all members](#)

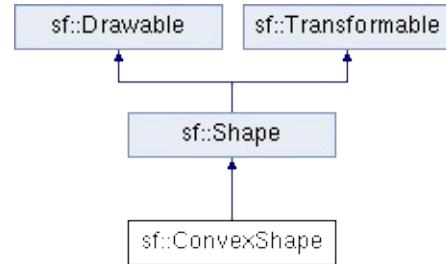
sf::ConvexShape Class Reference

Graphics module

Specialized shape representing a convex polygon. [More...](#)

```
#include <ConvexShape.hpp>
```

Inheritance diagram for sf::ConvexShape:



Public Member Functions

`ConvexShape (std::size_t pointCount=0)`
Default constructor. [More...](#)

`void setPointCount (std::size_t count)`
Set the number of points of the polygon. [More...](#)

`virtual std::size_t getPointCount () const`
Get the number of points of the polygon. [More...](#)

`void setPoint (std::size_t index, const Vector2f &point)`
Set the position of a point. [More...](#)

`virtual Vector2f getPoint (std::size_t index) const`
Get the position of a point. [More...](#)

`void setTexture (const Texture *texture, bool resetRect=false)`
Change the source texture of the shape. [More...](#)

`void setTextureRect (const IntRect &rect)`
Set the sub-rectangle of the texture that the shape will use.

`void setFillColor (const Color &color)`
Set the fill color of the shape. [More...](#)

`void setOutlineColor (const Color &color)`
Set the outline color of the shape. [More...](#)

`void setOutlineThickness (float thickness)`
Set the thickness of the shape's outline. [More...](#)

`const Texture * getTexture () const`

Get the source texture of the shape. [More...](#)

`const IntRect & getTextureRect () const`
Get the sub-rectangle of the texture displayed by the

`const Color & getFillColor () const`
Get the fill color of the shape. [More...](#)

`const Color & getOutlineColor () const`
Get the outline color of the shape. [More...](#)

`float getOutlineThickness () const`
Get the outline thickness of the shape. [More...](#)

`FloatRect getLocalBounds () const`
Get the local bounding rectangle of the entity. [More..](#)

`FloatRect getGlobalBounds () const`
Get the global (non-minimal) bounding rectangle of the

`void setPosition (float x, float y)`
set the position of the object [More...](#)

`void setPosition (const Vector2f &position)`
set the position of the object [More...](#)

`void setRotation (float angle)`
set the orientation of the object [More...](#)

`void setScale (float factorX, float factorY)`
set the scale factors of the object [More...](#)

`void setScale (const Vector2f &factors)`
set the scale factors of the object [More...](#)

`void setOrigin (float x, float y)`
set the local origin of the object [More...](#)

`void setOrigin (const Vector2f &origin)`
set the local origin of the object [More...](#)

`const Vector2f & getPosition () const`
get the position of the object [More...](#)

`float getRotation () const`
get the orientation of the object [More...](#)

`const Vector2f & getScale () const`
get the current scale of the object [More...](#)

`const Vector2f & getOrigin () const`
get the local origin of the object [More...](#)

`void move (float offsetX, float offsetY)`
Move the object by a given offset. [More...](#)

`void move (const Vector2f &offset)`
Move the object by a given offset. [More...](#)

`void rotate (float angle)`
Rotate the object. [More...](#)

`void scale (float factorX, float factorY)`
Scale the object. [More...](#)

`void scale (const Vector2f &factor)`
Scale the object. [More...](#)

`const Transform & getTransform () const`
get the combined transform of the object [More...](#)

```
const Transform & getInverseTransform () const
```

get the inverse of the combined transform of the object

Protected Member Functions

void `update ()`

Recompute the internal geometry of the shape. [More...](#)

Detailed Description

Specialized shape representing a convex polygon.

This class inherits all the functions of [sf::Transformable](#) (position, rotation functions of [sf::Shape](#) (outline, color, texture, ...)).

It is important to keep in mind that a convex shape must always be... drawn correctly. Moreover, the points must be defined in order; using a incorrect shape.

Usage example:

```
sf::ConvexShape polygon;
polygon.setPointCount(3);
polygon.setPoint(0, sf::Vector2f(0, 0));
polygon.setPoint(1, sf::Vector2f(0, 10));
polygon.setPoint(2, sf::Vector2f(25, 5));
polygon.setOutlineColor(sf::Color::Red);
polygon.setOutlineThickness(5);
polygon.setPosition(10, 20);
...
window.draw(polygon);
```

See also

[sf::Shape](#), [sf::RectangleShape](#), [sf::CircleShape](#)

Definition at line 42 of file [ConvexShape.hpp](#).

Constructor & Destructor Documentation

sf::ConvexShape::ConvexShape (std::size_t pointCount = 0)

Default constructor.

Parameters

pointCount Number of points of the polygon

Member Function Documentation

`const Color& sf::Shape::getFillColor() const`

Get the fill color of the shape.

Returns

Fill color of the shape

See also

[setFillColor](#)

`FloatRect sf::Shape::getGlobalBounds() const`

Get the global (non-minimal) bounding rectangle of the entity.

The returned rectangle is in global coordinates, which means it returns the bounds of the shape in the global 2D world's coordinate system. This function does not necessarily return the *minimal* bounding rectangle covers all the vertices (but possibly more). This allows bounds as a first check; you may want to use more precise checks on top of this.

Returns

Global bounding rectangle of the entity

const Transform& sf::Transformable::getInverseTransform() const

get the inverse of the combined transform of the object

Returns

Inverse of the combined transformations applied to the object

See also

[getTransform](#)

FloatRect sf::Shape::getLocalBounds() const

Get the local bounding rectangle of the entity.

The returned rectangle is in local coordinates, which means that (translation, rotation, scale, ...) that are applied to the entity. In other bounds of the entity in the entity's coordinate system.

Returns

Local bounding rectangle of the entity

const Vector2f& sf::Transformable::getOrigin() const

get the local origin of the object

Returns

Current origin

See also

[setOrigin](#)

const Color& sf::Shape::getOutlineColor() const

Get the outline color of the shape.

Returns

Outline color of the shape

See also

[setOutlineColor](#)

float sf::Shape::getOutlineThickness() const

Get the outline thickness of the shape.

Returns

Outline thickness of the shape

See also

[setOutlineThickness](#)

virtual Vector2f sf::ConvexShape::getPoint(std::size_t index) const

Get the position of a point.

The returned point is in local coordinates, that is, the shape's transforms taken into account. The result is undefined if *index* is out of the valid range [0 .. `getPointCount()` - 1].

Parameters

index Index of the point to get, in range [0 .. `getPointCount()` - 1]

Returns

Position of the index-th point of the polygon

See also

[setPoint](#)

Implements [sf::Shape](#).

virtual std::size_t sf::ConvexShape::getPointCount() const

Get the number of points of the polygon.

Returns

Number of points of the polygon

See also

[setPointCount](#)

Implements [sf::Shape](#).

const Vector2f& sf::Transformable::getPosition() const

get the position of the object

Returns

Current position

See also

[setPosition](#)

`float sf::Transformable::getRotation() const`

get the orientation of the object

The rotation is always in the range [0, 360].

Returns

Current rotation, in degrees

See also

[setRotation](#)

`const Vector2f& sf::Transformable::getScale() const`

get the current scale of the object

Returns

Current scale factors

See also

[setScale](#)

`const Texture* sf::Shape::getTexture() const`

Get the source texture of the shape.

If the shape has no source texture, a NULL pointer is returned. This means that you can't modify the texture when you retrieve it with this function.

Returns

Pointer to the shape's texture

See also

[setTexture](#)

const IntRect& sf::Shape::getTextureRect() const

Get the sub-rectangle of the texture displayed by the shape.

Returns

Texture rectangle of the shape

See also

[setTextureRect](#)

const Transform& sf::Transformable::getTransform() const

get the combined transform of the object

Returns

Transform combining the position/rotation/scale/origin of the object

See also

[getInverseTransform](#)

void sf::Transformable::move(float offsetX, float offsetY)

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition` equivalent to the following code:

```
sf::Vector2f pos = object.getPosition();
object.setPosition(pos.x + offsetX, pos.y + offsetY);
```

Parameters

offsetX X offset

offsetY Y offset

See also

[setPosition](#)

void sf::Transformable::move (const Vector2f & offset)

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition` equivalent to the following code:

```
object.setPosition(object.getPosition() + offset);
```

Parameters

offset Offset

See also

[setPosition](#)

void sf::Transformable::rotate (float angle)

Rotate the object.

This function adds to the current rotation of the object, unlike setRotation equivalent to the following code:

```
object.setRotation(object.getRotation() + angle);
```

Parameters

angle Angle of rotation, in degrees

```
void sf::Transformable::scale ( float factorX,  
                             float factorY  
                           )
```

Scale the object.

This function multiplies the current scale of the object, unlike setScale equivalent to the following code:

```
sf::Vector2f scale = object.getScale();  
object.setScale(scale.x * factorX, scale.y * factorY);
```

Parameters

factorX Horizontal scale factor

factorY Vertical scale factor

See also

[setScale](#)

```
void sf::Transformable::scale ( const Vector2f & factor )
```

Scale the object.

This function multiplies the current scale of the object, unlike `setScale` equivalent to the following code:

```
sf::Vector2f scale = object.getScale();
object.setScale(scale.x * factor.x, scale.y * factor.y);
```

Parameters

factor Scale factors

See also

`setScale`

void sf::Shape::setFillColor (const Color & color)

Set the fill color of the shape.

This color is modulated (multiplied) with the shape's texture if any. It can or change its global opacity. You can use `sf::Color::Transparent` to transparent, and have the outline alone. By default, the shape's fill color

Parameters

color New color of the shape

See also

`getFillColor`, `setOutlineColor`

void sf::Transformable::setOrigin (float x, float y)

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transform

Parameters

- x** X coordinate of the new origin
- y** Y coordinate of the new origin

See also

[getOrigin](#)

void sf::Transformable::setOrigin (const Vector2f & origin)

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transform

Parameters

- origin** New origin

See also

[getOrigin](#)

void sf::Shape::setOutlineColor (const Color & color)

Set the outline color of the shape.

By default, the shape's outline color is opaque white.

Parameters

color New outline color of the shape

See also

[getOutlineColor](#), [setFillColor](#)

void sf::Shape::setOutlineThickness (float thickness)

Set the thickness of the shape's outline.

Note that negative values are allowed (so that the outline expands toward the center). Using zero disables the outline. By default, the outline thickness is 0.

Parameters

thickness New outline thickness

See also

[getOutlineThickness](#)

void sf::ConvexShape::setPoint (std::size_t index, const Vector2f & point)

Set the position of a point.

Don't forget that the polygon must remain convex, and the points need to be called first in order to set the total number of points. The result must be valid range.

Parameters

index Index of the point to change, in range [0 .. [getPointCount\(\)](#) - 1]

point New position of the point

See also

[getPoint](#)

void sf::ConvexShape::setPointCount (std::size_t count)

Set the number of points of the polygon.

count must be greater than 2 to define a valid shape.

Parameters

count New number of points of the polygon

See also

[getPointCount](#)

**void sf::Transformable::setPosition (float x,
float y
)**

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

x X coordinate of the new position

y Y coordinate of the new position

See also

move, getPosition

void sf::Transformable::setPosition (const Vector2f & position)

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

position New position

See also

move, getPosition

void sf::Transformable::setRotation (float angle)

set the orientation of the object

This function completely overwrites the previous rotation. See the rotate on the previous rotation instead. The default rotation of a transformable c

Parameters

angle New rotation, in degrees

See also

rotate, getRotation

void sf::Transformable::setScale (float factorX, float factorY)

)

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function for more information. To keep the previous scale instead, use the `sf::Transformable::setScale` function.

Parameters

factorX New horizontal scale factor

factorY New vertical scale factor

See also

`scale`, `getScale`

void sf::Transformable::setScale (const Vector2f & factors)

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function for more information. To keep the previous scale instead, use the `sf::Transformable::setScale` function.

Parameters

factors New scale factors

See also

`scale`, `getScale`

**void sf::Shape::setTexture (const Texture * texture,
bool resetRect = false
)**

Change the source texture of the shape.

The *texture* argument refers to a texture that must exist as long as the shape does. It doesn't store its own copy of the texture, but rather keeps a pointer to it. If the source texture is destroyed and the shape tries to use it, the texture can be NULL to disable texturing. If *resetRect* is true, the TextureRect will be automatically adjusted to the size of the new texture. If it is false, the texture rect's width and height will be set to zero.

Parameters

texture New texture

resetRect Should the texture rect be reset to the size of the new texture?

See also

[getTexture](#), [setTextureRect](#)

void sf::Shape::setTextureRect (const IntRect & rect)

Set the sub-rectangle of the texture that the shape will display.

The texture rect is useful when you don't want to display the whole texture. By default, the texture rect covers the entire texture.

Parameters

rect Rectangle defining the region of the texture to display

See also

[getTextureRect](#), [setTexture](#)

void sf::Shape::update ()

Recompute the internal geometry of the shape.

This function must be called by the derived class everytime the shape's either getPointCount or getPoint is different).

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

- [ConvexShape.hpp](#)

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Member Functions](#) | [Protected Member Functions](#) | [Friends](#) | [List of all members](#)

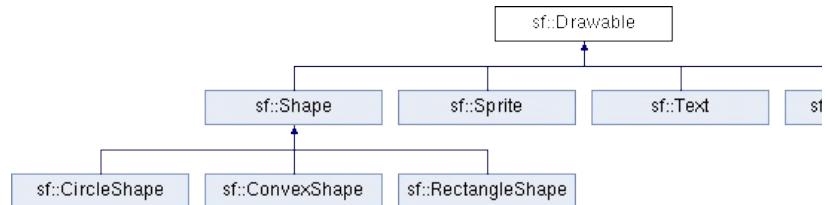
sf::Drawable Class Reference abstract

Graphics module

Abstract base class for objects that can be drawn to a render target. More

```
#include <Drawable.hpp>
```

Inheritance diagram for sf::Drawable:



Public Member Functions

virtual ~Drawable ()

Virtual destructor. [More...](#)

Protected Member Functions

```
virtual void draw(RenderTarget &target, RenderStates states) const =0  
    Draw the object to a render target. More...
```

Friends

class **RenderTarget**

Detailed Description

Abstract base class for objects that can be drawn to a render target.

`sf::Drawable` is a very simple base class that allows objects of derived classes to be drawn to a `sf::RenderTarget`.

All you have to do in your derived class is to override the `draw` virtual function.

Note that inheriting from `sf::Drawable` is not mandatory, but it allows this instead of calling `draw` directly rather than "object.draw(window)", which is more consistent with other SFML components.

Example:

```
class MyDrawable : public sf::Drawable
{
public:

    ...

private:
    virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
    {
        // You can draw other high-level objects
        target.draw(m_sprite, states);

        // ... or use the low-level API
        states.texture = &m_texture;
        target.draw(m_vertices, states);

        // ... or draw with OpenGL directly
        glBegin(GL_QUADS);
        ...
        glEnd();
    }

    sf::Sprite m_sprite;
    sf::Texture m_texture;
    sf::VertexArray m_vertices;
};
```

See also

[sf::RenderTarget](#)

Definition at line 44 of file [Drawable.hpp](#).

Constructor & Destructor Documentation

virtual sf::Drawable::~Drawable()

Virtual destructor.

Definition at line 52 of file [Drawable.hpp](#).

Member Function Documentation

```
virtual void sf::Drawable::draw( RenderTarget & target,  
                                RenderStates states  
) const
```

Draw the object to a render target.

This is a pure virtual function that has to be implemented by the derived class to draw the object. It takes a render target and current render states as parameters.

Parameters

target Render target to draw to
states Current render states

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

- [Drawable.hpp](#)

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Classes](#) | [Public Member Functions](#) | [List of all members](#)

sf::Font Class Reference

Graphics module

Class for loading and manipulating character fonts. [More...](#)

```
#include <Font.hpp>
```

Classes

struct [Info](#)

Holds various information about a font. [More...](#)

Public Member Functions

`Font ()`

Default constructor. [More...](#)

`Font (const Font ©)`

Copy constructor. [More...](#)

`~Font ()`

Destructor. [More...](#)

`bool loadFromFile (const std::string &filename)`

Load the font from a file. [More...](#)

`bool loadFromMemory (const void *data, std::size_t sizeInByt`

Load the font from a file in memory. [More...](#)

`bool loadFromStream (InputStream &stream)`

Load the font from a custom stream. [More...](#)

`const Info & getInfo () const`

Get the font information. [More...](#)

`const Glyph & getGlyph (UInt32 codePoint, unsigned int characterSize`

`outlineThickness=0) const`

Retrieve a glyph of the font. [More...](#)

`float getKerning (UInt32 first, UInt32 second, unsigned int ch`

Get the kerning offset of two glyphs. [More...](#)

`float getLineSpacing (unsigned int characterSize) const`

Get the line spacing. [More...](#)

float `getUnderlinePosition` (unsigned int characterSize) const
Get the position of the underline. [More...](#)

float `getUnderlineThickness` (unsigned int characterSize) const
Get the thickness of the underline. [More...](#)

const Texture & `getTexture` (unsigned int characterSize) const
Retrieve the texture containing the loaded glyphs of a c

Font & `operator=` (const Font &right)
Overload of assignment operator. [More...](#)

Detailed Description

Class for loading and manipulating character fonts.

Fonts can be loaded from a file, from memory or from a custom stream types of fonts.

See the `loadFromFile` function for the complete list of supported formats.

Once it is loaded, a `sf::Font` instance provides three types of information ↴

- Global metrics, such as the line spacing
- Per-glyph metrics, such as bounding box or kerning
- Pixel representation of glyphs

Fonts alone are not very useful: they hold the font data but cannot make ↴ need to use the `sf::Text` class, which is able to properly output text with ↴ size, style, color, position, rotation, etc. This separation allows more flexibility indeed a `sf::Font` is a heavy resource, and any operation on it is slow (in applications). On the other side, a `sf::Text` is a lightweight object which can use metrics of a `sf::Font` to display any text on a render target. Note that it is possible to associate multiple `sf::Text` instances to the same `sf::Font`.

It is important to note that the `sf::Text` instance doesn't copy the font that it uses to it. Thus, a `sf::Font` must not be destructed while it is used by a `sf::Text` (the `sf::Text` uses a local `sf::Font` instance for creating a text).

Usage example:

```
// Declare a new font
sf::Font font;
```

```
// Load it from a file
if (!font.loadFromFile("arial.ttf"))
{
    // error...
}

// Create a text which uses our font
sf::Text text1;
text1.setFont(font);
text1.setCharacterSize(30);
text1.setStyle(sf::Text::Regular);

// Create another text using the same font, but with different parameters
sf::Text text2;
text2.setFont(font);
text2.setCharacterSize(50);
text2.setStyle(sf::Text::Italic);
```

Apart from loading font files, and passing them to instances of `sf::Text`, deal directly with this class. However, it may be useful to access the font for advanced usage.

Note that if the font is a bitmap font, it is not scalable, thus not all requested sizes will be available. This needs to be taken into consideration when using `sf::Text`. If you need a specific size, make sure the corresponding bitmap font that supports that size is used.

See also

`sf::Text`

Definition at line 50 of file `Font.hpp`.

Constructor & Destructor Documentation

sf::Font::Font()

Default constructor.

This constructor defines an empty font

sf::Font::Font(const Font & **copy)**

Copy constructor.

Parameters

copy Instance to copy

sf::Font::~Font()

Destructor.

Cleans up all the internal resources used by the font

Member Function Documentation

```
const Glyph& sf::Font::getGlyph ( Uint32 codePoint,  
                                unsigned int characterSize,  
                                bool bold,  
                                float outlineThickness =  
                                ) const
```

Retrieve a glyph of the font.

If the font is a bitmap font, not all character sizes might be available. If requested size, an empty glyph is returned.

Be aware that using a negative value for the outline thickness will cause

Parameters

codePoint	Unicode code point of the character to get
characterSize	Reference character size
bold	Retrieve the bold version or the regular one?
outlineThickness	Thickness of outline (when != 0 the glyph will not

Returns

The glyph corresponding to *codePoint* and *characterSize*

```
const Info& sf::Font:: getInfo ( ) const
```

Get the font information.

Returns

A structure that holds the font information

```
float sf::Font::getKerning ( Uint32 first,  
                           Uint32 second,  
                           unsigned int characterSize  
                         ) const
```

Get the kerning offset of two glyphs.

The kerning is an extra offset (negative) to apply between two glyphs which will pair look more "natural". For example, the pair "AV" have a special kerning other characters. Most of the glyphs pairs have a kerning offset of zero, though.

Parameters

first Unicode code point of the first character
second Unicode code point of the second character
characterSize Reference character size

Returns

Kerning value for *first* and *second*, in pixels

```
float sf::Font::getLineSpacing ( unsigned int characterSize ) const
```

Get the line spacing.

Line spacing is the vertical offset to apply between two consecutive lines.

Parameters

characterSize Reference character size

Returns

Line spacing, in pixels

const **Texture&** sf::Font::getTexture (unsigned int characterSize) const

Retrieve the texture containing the loaded glyphs of a certain size.

The contents of the returned texture changes as more glyphs are requested.
It is mainly used internally by sf::Text.

Parameters

characterSize Reference character size

Returns

Texture containing the glyphs of the requested size

float sf::Font::getUnderlinePosition (unsigned int characterSize) const

Get the position of the underline.

Underline position is the vertical offset to apply between the baseline and the underline.

Parameters

characterSize Reference character size

Returns

Underline position, in pixels

See also

[getUnderlineThickness](#)

float sf::Font::getUnderlineThickness (unsigned int `characterSize`)

Get the thickness of the underline.

Underline thickness is the vertical size of the underline.

Parameters

characterSize Reference character size

Returns

Underline thickness, in pixels

See also

[getUnderlinePosition](#)

bool sf::Font::loadFromFile (const std::string & `filename`)

Load the font from a file.

The supported font formats are: TrueType, Type 1, CFF, OpenType, BDF, PFR and Type 42. Note that this function know nothing about the user's system, thus you can't load them directly.

Warning

SFML cannot preload all the font data in this function, so the file has to be loaded again if the [sf::Font](#) object loads a new font or is destroyed.

Parameters

filename Path of the font file to load

Returns

True if loading succeeded, false if it failed

See also

[loadFromMemory](#), [loadFromStream](#)

```
bool sf::Font::loadFromMemory ( const void * data,
                               std::size_t sizeInBytes
                             )
```

Load the font from a file in memory.

The supported font formats are: TrueType, Type 1, CFF, OpenType, Type 1C, BDF, PFR and Type 42.

Warning

SFML cannot preload all the font data in this function, so the buffer passed to the function is valid until the [sf::Font](#) object loads a new font or is destroyed.

Parameters

data Pointer to the file data in memory
sizeInBytes Size of the data to load, in bytes

Returns

True if loading succeeded, false if it failed

See also

[loadFromFile](#), [loadFromStream](#)

```
bool sf::Font::loadFromStream ( InputStream & stream )
```

Load the font from a custom stream.

The supported font formats are: TrueType, Type 1, CFF, OpenType, BDF, PFR and Type 42. Warning: SFML cannot preload all the font data of *stream* have to remain valid as long as the font is used.

Warning

SFML cannot preload all the font data in this function, so the stream the `sf::Font` object loads a new font or is destroyed.

Parameters

stream Source stream to read from

Returns

True if loading succeeded, false if it failed

See also

`loadFromFile`, `loadFromMemory`

`Font& sf::Font::operator= (const Font & right)`

Overload of assignment operator.

Parameters

right Instance to assign

Returns

Reference to self

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

- `Font.hpp`

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Attributes](#) | [List of all members](#)

sf::Font::Info Struct Reference

Holds various information about a font. [More...](#)

```
#include <Font.hpp>
```

Public Attributes

std::string **family**

The font family. More...

Detailed Description

Holds various information about a font.

Definition at line [58](#) of file [Font.hpp](#).

Member Data Documentation

std::string sf::Font::Info::family

The font family.

Definition at line **60** of file [Font.hpp](#).

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

- [Font.hpp](#)
-

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [Public Attributes](#) | [List of all members](#)

sf::Glyph Class Reference

[Graphics module](#)

Structure describing a glyph. More...

```
#include <Glyph.hpp>
```

Public Member Functions

[Glyph \(\)](#)

Default constructor. More...

Public Attributes

float `advance`

Offset to move horizontally to the next character. More...

`FloatRect bounds`

Bounding rectangle of the glyph, in coordinates relative to the

`IntRect textureRect`

Texture coordinates of the glyph inside the font's texture. Mor

Detailed Description

Structure describing a glyph.

A glyph is the visual representation of a character.

The `sf::Glyph` structure provides the information needed to handle the gly|

- its coordinates in the font's texture
- its bounding rectangle
- the offset to apply to get the starting position of the next glyph

See also

`sf::Font`

Definition at line 41 of file `Glyph.hpp`.

Constructor & Destructor Documentation

sf::Glyph::Glyph()

Default constructor.

Definition at line [49](#) of file [Glyph.hpp](#).

Member Data Documentation

float sf::Glyph::advance

Offset to move horizontally to the next character.

Definition at line 54 of file [Glyph.hpp](#).

FloatRect sf::Glyph::bounds

Bounding rectangle of the glyph, in coordinates relative to the baseline.

Definition at line 55 of file [Glyph.hpp](#).

IntRect sf::Glyph::textureRect

Texture coordinates of the glyph inside the font's texture.

Definition at line 56 of file [Glyph.hpp](#).

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

- [Glyph.hpp](#)
-

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Member Functions](#) | [List of all members](#)

sf::Image Class Reference

Graphics module

Class for loading, manipulating and saving images. [More...](#)

#include <Image.hpp>

Public Member Functions

[Image \(\)](#)

Default constructor. [More...](#)

[~Image \(\)](#)

Destructor. [More...](#)

`void create (unsigned int width, unsigned int height, const Color & color)`
Create the image and fill it with a unique color. [More...](#)

`void create (unsigned int width, unsigned int height, const Uint8 * pixels)`
Create the image from an array of pixels. [More...](#)

`bool loadFromFile (const std::string &filename)`
Load the image from a file on disk. [More...](#)

`bool loadFromMemory (const void *data, std::size_t size)`
Load the image from a file in memory. [More...](#)

`bool loadFromStream (InputStream &stream)`
Load the image from a custom stream. [More...](#)

`bool saveToFile (const std::string &filename) const`
Save the image to a file on disk. [More...](#)

`Vector2u getSize () const`

Return the size (width and height) of the image. [More...](#)

`void createMaskFromColor (const Color &color, Uint8 alpha=0)`
Create a transparency mask from a specified color-key. [More...](#)

`copy (const Image &source, unsigned int destX, unsigned int destY, unsigned int width, unsigned int height)`

```
void &sourceRect=IntRect(0, 0, 0, 0), bool applyAlpha=false)
```

Copy pixels from another image onto this one. [More...](#)

```
void setPixel (unsigned int x, unsigned int y, const Color &color)
```

Change the color of a pixel. [More...](#)

```
Color getPixel (unsigned int x, unsigned int y) const
```

Get the color of a pixel. [More...](#)

```
const Uint8 * getPixelsPtr () const
```

Get a read-only pointer to the array of pixels. [More...](#)

```
void flipHorizontally ()
```

Flip the image horizontally (left <-> right) [More...](#)

```
void flipVertically ()
```

Flip the image vertically (top <-> bottom) [More...](#)

Detailed Description

Class for loading, manipulating and saving images.

`sf::Image` is an abstraction to manipulate images as bidimensional arrays

The class provides functions to load, read, write and save pixels, as well as

`sf::Image` can handle a unique internal representation of pixels, which is pixel must be composed of 8 bits red, green, blue and alpha channel. All functions that return an array of pixels follow this rule, and all parameters of functions (such as `loadFromMemory`) must use this representation as well.

A `sf::Image` can be copied, but it is a heavy resource and if possible references to pass or return them to avoid useless copies.

Usage example:

```
// Load an image file from a file
sf::Image background;
if (!background.loadFromFile("background.jpg"))
    return -1;

// Create a 20x20 image filled with black color
sf::Image image;
image.create(20, 20, sf::Color::Black);

// Copy image1 on image2 at position (10, 10)
image.copy(background, 10, 10);

// Make the top-left pixel transparent
sf::Color color = image.getPixel(0, 0);
color.a = 0;
image.setPixel(0, 0, color);

// Save the image to a file
if (!image.saveToFile("result.png"))
    return -1;
```

See also

[sf::Texture](#)

Definition at line 46 of file [Image.hpp](#).

Constructor & Destructor Documentation

sf::Image::Image ()

Default constructor.

Creates an empty image.

sf::Image::~Image ()

Destructor.

Member Function Documentation

```
void sf::Image::copy ( const Image & source,
                      unsigned int      destX,
                      unsigned int      destY,
                      const IntRect & sourceRect = IntRect(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 to copy complex static image from several others, but if you need this kind of feature, use `sf::RenderTexture`.

If `sourceRect` is empty, the whole image is copied. If `applyAlpha` is set to `true`, alpha pixels are applied. If it is `false`, the pixels are copied unchanged with their original colors.

Parameters

<code>source</code>	Source image to copy
<code>destX</code>	X coordinate of the destination position
<code>destY</code>	Y coordinate of the destination position
<code>sourceRect</code>	Sub-rectangle of the source image to copy
<code>applyAlpha</code>	Should the copy take into account the source transparency?

```
void sf::Image::create ( unsigned int   width,
                       unsigned int   height,
                       const Color & color = Color(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 same width and height as the image. If *pixels* is null, an empty image is created. If this is an undefined behavior. 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 & 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 zero, making them transparent.

Parameters

color Color 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 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 undefined behavior.

Parameters

x X coordinate of pixel to get

y Y coordinate of pixel to get

Returns

`Color` of the pixel at coordinates (x, y)

See also

`setPixel`

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 in the array is width * height * 4 (`getSize().x * getSize().y * 4`). Warning: this is invalid if you modify the image, so you should never store it for too long. The pointer is returned.

Returns

Read-only pointer to the array of pixels

Vector2u sf::Image::getSize() const

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

Returns

Size 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 pvr.

supported, like progressive jpeg. If this function fails, the image is left unchanged.

Parameters

filename Path of the image file to load

Returns

True if loading was successful

See also

[loadFromMemory](#), [loadFromStream](#), [saveToFile](#)

```
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 psb. Other formats 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

Returns

True if loading was successful

See also

[loadFromFile](#), [loadFromStream](#)

```
bool sf::Image::loadFromStream ( InputStream & stream )
```

Load the image from a custom stream.

The supported image formats are bmp, png, tga, jpg, gif, psd, hdr and psb. Some formats are supported, like progressive jpeg. If this function fails, the image is left unchanged.

Parameters

stream Source stream to read from

Returns

True if loading was successful

See also

[loadFromFile](#), [loadFromMemory](#)

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 formats are bmp, png, tga and jpg. The destination file is overwritten if it already exists. The image is saved even if it is empty.

Parameters

filename Path of the file to save

Returns

True if saving was successful

See also

[create](#), [loadFromFile](#), [loadFromMemory](#)

```
void sf::Image::setPixel ( unsigned int x,
                         unsigned int y,
                         const Color & color
)
```

Change the color of a pixel.

This function doesn't check the validity of the pixel coordinates, using undefined behavior.

Parameters

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

See also

[getPixel](#)

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

- [Image.hpp](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [Public Attributes](#) | [Related Functions](#) | [List of all members](#)

sf::Rect< T > Class Template Reference

[Graphics module](#)

Utility class for manipulating 2D axis aligned rectangles. More...

```
#include <Rect.hpp>
```

Public Member Functions

`Rect ()`

Default constructor. [More...](#)

`Rect (T rectLeft, T rectTop, T rectWidth, T rectHeight)`

Construct the rectangle from its coordinates. [More...](#)

`Rect (const Vector2< T > &position, const Vector2< T > &size)`

Construct the rectangle from position and size. [More...](#)

`template<typename U >`

`Rect (const Rect< U > &rectangle)`

Construct the rectangle from another type of rectangle. [More...](#)

`bool contains (T x, T y) const`

Check if a point is inside the rectangle's area. [More...](#)

`bool contains (const Vector2< T > &point) const`

Check if a point is inside the rectangle's area. [More...](#)

`bool intersects (const Rect< T > &rectangle) const`

Check the intersection between two rectangles. [More...](#)

`bool intersects (const Rect< T > &rectangle, Rect< T > &intersection) const`

Check the intersection between two rectangles. [More...](#)

Public Attributes

T `left`

Left coordinate of the rectangle. [More...](#)

T `top`

Top coordinate of the rectangle. [More...](#)

T `width`

Width of the rectangle. [More...](#)

T `height`

Height of the rectangle. [More...](#)

Related Functions

(Note that these are not member functions.)

```
template<typename T >
bool operator==(const Rect< T > &left, const Rect< T > &right)
    Overload of binary operator ==. More...
```

```
template<typename T >
bool operator!=(const Rect< T > &left, const Rect< T > &right)
    Overload of binary operator !=. More...
```

Detailed Description

```
template<typename T>
class sf::Rect< T >
```

Utility class for manipulating 2D axis aligned rectangles.

A rectangle is defined by its top-left corner and its size.

It is a very simple class defined for convenience, so its member variable public and can be accessed directly, just like the vector classes ([Vector2](#) &

To keep things simple, `sf::Rect` doesn't define functions to emulate the members (such as right, bottom, center, etc.), it rather only provides inters

`sf::Rect` uses the usual rules for its boundaries:

- The left and top edges are included in the rectangle's area
- The right (left + width) and bottom (top + height) edges are excluded fr

This means that `sf::IntRect(0, 0, 1, 1)` and `sf::IntRect(1, 1, 1, 1)` don't intersect.

`sf::Rect` is a template and may be used with any numeric type, but for simplicity SFML are typedef'd:

- `sf::Rect<int>` is `sf::IntRect`
- `sf::Rect<float>` is `sf::FloatRect`

So that you don't have to care about the template syntax.

Usage example:

```
// Define a rectangle, located at (0, 0) with a size of 20x5
sf::IntRect r1(0, 0, 20, 5);

// Define another rectangle, located at (4, 2) with a size of 18x10
sf::Vector2i position(4, 2);
sf::Vector2i size(18, 10);
sf::IntRect r2(position, size);

// Test intersections with the point (3, 1)
bool b1 = r1.contains(3, 1); // true
bool b2 = r2.contains(3, 1); // false

// Test the intersection between r1 and r2
sf::IntRect result;
bool b3 = r1.intersects(r2, result); // true
// result == (4, 2, 16, 3)
```

Definition at line 42 of file [Rect.hpp](#).

Constructor & Destructor Documentation

```
template<typename T>
sf::Rect< T >::Rect()
```

Default constructor.

Creates an empty rectangle (it is equivalent to calling Rect(0, 0, 0, 0)).

```
template<typename T>
sf::Rect< T >::Rect(T rectLeft,
                     T rectTop,
                     T rectWidth,
                     T rectHeight
)
```

Construct the rectangle from its coordinates.

Be careful, the last two parameters are the width and height, not the right and bottom edges.

Parameters

- rectLeft** Left coordinate of the rectangle
- rectTop** Top coordinate of the rectangle
- rectWidth** Width of the rectangle
- rectHeight** Height of the rectangle

```
template<typename T>
sf::Rect< T >::Rect ( const Vector2< T > & position,
                      const Vector2< T > & size
                    )
```

Construct the rectangle from position and size.

Be careful, the last parameter is the size, not the bottom-right corner!

Parameters

position Position of the top-left corner of the rectangle
size Size of the rectangle

```
template<typename T>
template<typename U >
sf::Rect< T >::Rect ( const Rect< U > & rectangle )
```

Construct the rectangle from another type of rectangle.

This constructor doesn't replace the copy constructor, it's called on constructor will fail to compile if U is not convertible to T.

Parameters

rectangle Rectangle to convert

Member Function Documentation

```
template<typename T>
bool sf::Rect< T >::contains ( T x,
                               T y
                           ) const
```

Check if a point is inside the rectangle's area.

This check is non-inclusive. If the point lies on the edge of the rectangle,

Parameters

- x** X coordinate of the point to test
- y** Y coordinate of the point to test

Returns

True if the point is inside, false otherwise

See also

[intersects](#)

```
template<typename T>
bool sf::Rect< T >::contains ( const Vector2< T > & point ) const
```

Check if a point is inside the rectangle's area.

This check is non-inclusive. If the point lies on the edge of the rectangle,

Parameters

point Point to test

Returns

True if the point is inside, false otherwise

See also

[intersects](#)

```
template<typename T>
```

```
bool sf::Rect< T >::intersects ( const Rect< T > & rectangle ) const
```

Check the intersection between two rectangles.

Parameters

rectangle Rectangle to test

Returns

True if rectangles overlap, false otherwise

See also

[contains](#)

```
template<typename T>
```

```
bool sf::Rect< T >::intersects ( const Rect< T > & rectangle,
                                  Rect< T > & intersection
                                )
                                const
```

Check the intersection between two rectangles.

This overload returns the overlapped rectangle in the *intersection* parameter.

Parameters

rectangle Rectangle to test

intersection Rectangle to be filled with the intersection

Returns

True if rectangles overlap, false otherwise

See also

[contains](#)

Friends And Related Function Documentation

```
template<typename T >
bool operator!= ( const Rect< T > & left,
                    const Rect< T > & right
                )
```

Overload of binary operator !=.

This operator compares strict difference between two rectangles.

Parameters

left Left operand (a rectangle)
right Right operand (a rectangle)

Returns

True if *left* is not equal to *right*

```
template<typename T >
bool operator== ( const Rect< T > & left,
                    const Rect< T > & right
                )
```

Overload of binary operator ==.

This operator compares strict equality between two rectangles.

Parameters

left Left operand (a rectangle)

right Right operand (a rectangle)

Returns

True if *left* is equal to *right*

Member Data Documentation

template<typename T>

T sf::Rect< T >::height

Height of the rectangle.

Definition at line 160 of file Rect.hpp.

template<typename T>

T sf::Rect< T >::left

Left coordinate of the rectangle.

Definition at line 157 of file Rect.hpp.

template<typename T>

T sf::Rect< T >::top

Top coordinate of the rectangle.

Definition at line 158 of file Rect.hpp.

template<typename T>

T sf::Rect< T >::width

Width of the rectangle.

Definition at line 159 of file Rect.hpp.

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

- [Rect.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#).

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Protected Member Functions](#) | [List of all members](#)

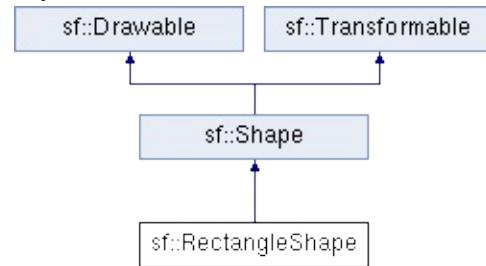
sf::RectangleShape Class Reference

Graphics module

Specialized shape representing a rectangle. [More...](#)

```
#include <RectangleShape.hpp>
```

Inheritance diagram for sf::RectangleShape:



Public Member Functions

`RectangleShape (const Vector2f &size=Vector2f(0, 0))`
Default constructor. [More...](#)

`void setSize (const Vector2f &size)`
Set the size of the rectangle. [More...](#)

`const Vector2f & getSize () const`
Get the size of the rectangle. [More...](#)

`virtual std::size_t getPointCount () const`
Get the number of points defining the shape. [More...](#)

`virtual Vector2f getPoint (std::size_t index) const`
Get a point of the rectangle. [More...](#)

`void setTexture (const Texture *texture, bool resetRect=false)`
Change the source texture of the shape. [More...](#)

`void setTextureRect (const IntRect &rect)`
Set the sub-rectangle of the texture that the shape will use.

`void setFillColor (const Color &color)`
Set the fill color of the shape. [More...](#)

`void setOutlineColor (const Color &color)`
Set the outline color of the shape. [More...](#)

`void setOutlineThickness (float thickness)`
Set the thickness of the shape's outline. [More...](#)

`const Texture * getTexture () const`

Get the source texture of the shape. [More...](#)

`const IntRect & getTextureRect () const`
Get the sub-rectangle of the texture displayed by the

`const Color & getFillColor () const`
Get the fill color of the shape. [More...](#)

`const Color & getOutlineColor () const`
Get the outline color of the shape. [More...](#)

`float getOutlineThickness () const`
Get the outline thickness of the shape. [More...](#)

`FloatRect getLocalBounds () const`
Get the local bounding rectangle of the entity. [More..](#)

`FloatRect getGlobalBounds () const`
Get the global (non-minimal) bounding rectangle of the

`void setPosition (float x, float y)`
set the position of the object [More...](#)

`void setPosition (const Vector2f &position)`
set the position of the object [More...](#)

`void setRotation (float angle)`
set the orientation of the object [More...](#)

`void setScale (float factorX, float factorY)`
set the scale factors of the object [More...](#)

`void setScale (const Vector2f &factors)`
set the scale factors of the object [More...](#)

`void setOrigin (float x, float y)`
set the local origin of the object [More...](#)

`void setOrigin (const Vector2f &origin)`
set the local origin of the object [More...](#)

`const Vector2f & getPosition () const`
get the position of the object [More...](#)

`float getRotation () const`
get the orientation of the object [More...](#)

`const Vector2f & getScale () const`
get the current scale of the object [More...](#)

`const Vector2f & getOrigin () const`
get the local origin of the object [More...](#)

`void move (float offsetX, float offsetY)`
Move the object by a given offset. [More...](#)

`void move (const Vector2f &offset)`
Move the object by a given offset. [More...](#)

`void rotate (float angle)`
Rotate the object. [More...](#)

`void scale (float factorX, float factorY)`
Scale the object. [More...](#)

`void scale (const Vector2f &factor)`
Scale the object. [More...](#)

`const Transform & getTransform () const`
get the combined transform of the object [More...](#)

```
const Transform & getInverseTransform () const
```

get the inverse of the combined transform of the object

Protected Member Functions

void `update ()`

Recompute the internal geometry of the shape. [More...](#)

Detailed Description

Specialized shape representing a rectangle.

This class inherits all the functions of [sf::Transformable](#) (position, rotation functions of [sf::Shape](#) (outline, color, texture, ...).

Usage example:

```
sf::RectangleShape rectangle;
rectangle.setSize(sf::Vector2f(100, 50));
rectangle.setOutlineColor(sf::Color::Red);
rectangle.setOutlineThickness(5);
rectangle.setPosition(10, 20);
...
window.draw(rectangle);
```

See also

[sf::Shape](#), [sf::CircleShape](#), [sf::ConvexShape](#)

Definition at line 41 of file [RectangleShape.hpp](#).

Constructor & Destructor Documentation

```
sf::RectangleShape::RectangleShape ( const Vector2f & size = Vector2f(0, 0) )
```

Default constructor.

Parameters

size Size of the rectangle

Member Function Documentation

`const Color& sf::Shape::getFillColor() const`

Get the fill color of the shape.

Returns

Fill color of the shape

See also

[setFillColor](#)

`FloatRect sf::Shape::getGlobalBounds() const`

Get the global (non-minimal) bounding rectangle of the entity.

The returned rectangle is in global coordinates, which means it returns the bounds of the shape in the global 2D world's coordinate system. This function does not necessarily return the *minimal* bounding rectangle covers all the vertices (but possibly more). This allows bounds as a first check; you may want to use more precise checks on top of this.

Returns

Global bounding rectangle of the entity

const Transform& sf::Transformable::getInverseTransform() const

get the inverse of the combined transform of the object

Returns

Inverse of the combined transformations applied to the object

See also

[getTransform](#)

FloatRect sf::Shape::getLocalBounds() const

Get the local bounding rectangle of the entity.

The returned rectangle is in local coordinates, which means that (translation, rotation, scale, ...) that are applied to the entity. In other bounds of the entity in the entity's coordinate system.

Returns

Local bounding rectangle of the entity

const Vector2f& sf::Transformable::getOrigin() const

get the local origin of the object

Returns

Current origin

See also

[setOrigin](#)

const Color& sf::Shape::getOutlineColor() const

Get the outline color of the shape.

Returns

Outline color of the shape

See also

[setOutlineColor](#)

float sf::Shape::getOutlineThickness() const

Get the outline thickness of the shape.

Returns

Outline thickness of the shape

See also

[setOutlineThickness](#)

virtual Vector2f sf::RectangleShape::getPoint(std::size_t index) const

Get a point of the rectangle.

The returned point is in local coordinates, that is, the shape's transforms taken into account. The result is undefined if *index* is out of the valid range [0 .. 3].

Parameters

index Index of the point to get, in range [0 .. 3]

Returns

index-th point of the shape

Implements [sf::Shape](#).

virtual std::size_t sf::RectangleShape::getPointCount() const

Get the number of points defining the shape.

Returns

Number of points of the shape. For rectangle shapes, this number is

Implements [sf::Shape](#).

const Vector2f& sf::Transformable::getPosition() const

get the position of the object

Returns

Current position

See also

[setPosition](#)

float sf::Transformable::getRotation() const

get the orientation of the object

The rotation is always in the range [0, 360].

Returns

Current rotation, in degrees

See also

[setRotation](#)

const `Vector2f&` sf::Transformable::getScale () const

get the current scale of the object

Returns

Current scale factors

See also

[setScale](#)

const `Vector2f&` sf::RectangleShape::getSize () const

Get the size of the rectangle.

Returns

Size of the rectangle

See also

[setSize](#)

const `Texture*` sf::Shape::getTexture () const

Get the source texture of the shape.

If the shape has no source texture, a NULL pointer is returned. This means that you can't modify the texture when you retrieve it with this function.

Returns

Pointer to the shape's texture

See also

[setTexture](#)

const IntRect& sf::Shape::getTextureRect() const

Get the sub-rectangle of the texture displayed by the shape.

Returns

Texture rectangle of the shape

See also

[setTextureRect](#)

const Transform& sf::Transformable::getTransform() const

get the combined transform of the object

Returns

Transform combining the position/rotation/scale/origin of the object

See also

[getInverseTransform](#)

void sf::Transformable::move(float offsetX,

```
    float offsetY  
)  
}
```

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition` equivalent to the following code:

```
sf::Vector2f pos = object.getPosition();  
object.setPosition(pos.x + offsetX, pos.y + offsetY);
```

Parameters

offsetX X offset
offsetY Y offset

See also

[setPosition](#)

void sf::Transformable::move (const Vector2f & offset)

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition` equivalent to the following code:

```
object.setPosition(object.getPosition() + offset);
```

Parameters

offset Offset

See also

[setPosition](#)

void sf::Transformable::rotate (float angle)

Rotate the object.

This function adds to the current rotation of the object, unlike setRotation equivalent to the following code:

```
object.setRotation(object.getRotation() + angle);
```

Parameters

angle Angle of rotation, in degrees

void sf::Transformable::scale (float factorX, float factorY)

Scale the object.

This function multiplies the current scale of the object, unlike setScale equivalent to the following code:

```
sf::Vector2f scale = object.getScale();  
object.setScale(scale.x * factorX, scale.y * factorY);
```

Parameters

factorX Horizontal scale factor

factorY Vertical scale factor

See also

[setScale](#)

void sf::Transformable::scale (const Vector2f & factor)

Scale the object.

This function multiplies the current scale of the object, unlike setScale equivalent to the following code:

```
sf::Vector2f scale = object.getScale();
object.setScale(scale.x * factor.x, scale.y * factor.y);
```

Parameters

factor Scale factors

See also

[setScale](#)

void sf::Shape::setFillColor (const Color & color)

Set the fill color of the shape.

This color is modulated (multiplied) with the shape's texture if any. It can or change its global opacity. You can use [sf::Color::Transparent](#) to transparent, and have the outline alone. By default, the shape's fill color

Parameters

color New color of the shape

See also

[getFillColor](#), [setOutlineColor](#)

```
void sf::Transformable::setOrigin ( float x,  
                                    float y  
                                )
```

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transfor

Parameters

- x** X coordinate of the new origin
- y** Y coordinate of the new origin

See also

[getOrigin](#)

```
void sf::Transformable::setOrigin ( const Vector2f & origin )
```

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transfor

Parameters

- origin** New origin

See also

[getOrigin](#)

void sf::Shape::setOutlineColor (const Color & color)

Set the outline color of the shape.

By default, the shape's outline color is opaque white.

Parameters

color New outline color of the shape

See also

[getOutlineColor](#), [setFillColor](#)

void sf::Shape::setOutlineThickness (float thickness)

Set the thickness of the shape's outline.

Note that negative values are allowed (so that the outline expands toward the center). Using zero disables the outline. By default, the outline thickness is 0.

Parameters

thickness New outline thickness

See also

[getOutlineThickness](#)

void sf::Transformable::setPosition (float x, float y)

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

- x** X coordinate of the new position
- y** Y coordinate of the new position

See also

[move](#), [getPosition](#)

void sf::Transformable::setPosition (const Vector2f & position)

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

- position** New position

See also

[move](#), [getPosition](#)

void sf::Transformable::setRotation (float angle)

set the orientation of the object

This function completely overwrites the previous rotation. See the rotate on the previous rotation instead. The default rotation of a transformable

Parameters

angle New rotation, in degrees

See also

[rotate](#), [getRotation](#)

```
void sf::Transformable::setScale ( float factorX,  
                                 float factorY  
 )
```

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function instead. The default scale of a transformable object is (1, 1).

Parameters

factorX New horizontal scale factor
factorY New vertical scale factor

See also

[scale](#), [getScale](#)

```
void sf::Transformable::setScale ( const Vector2f & factors )
```

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function instead. The default scale of a transformable object is (1, 1).

Parameters

factors New scale factors

See also

[scale](#), [getScale](#)

void sf::RectangleShape::setSize (const Vector2f & size)

Set the size of the rectangle.

Parameters

size New size of the rectangle

See also

[getSize](#)

void sf::Shape::setTexture (const Texture * texture, bool resetRect = false)

Change the source texture of the shape.

The *texture* argument refers to a texture that must exist as long as the shape does. It doesn't store its own copy of the texture, but rather keeps a pointer to it. If the source texture is destroyed and the shape tries to use it, the texture can be NULL to disable texturing. If *resetRect* is true, the TextureRect will be automatically adjusted to the size of the new texture. If it is false, the texture rect's width and height will be set to zero.

Parameters

texture New texture

resetRect Should the texture rect be reset to the size of the new texture?

See also

[getTexture](#), [setTextureRect](#)

void sf::Shape::setTextureRect (const IntRect & rect)

Set the sub-rectangle of the texture that the shape will display.

The texture rect is useful when you don't want to display the whole texture by default, the texture rect covers the entire texture.

Parameters

rect Rectangle defining the region of the texture to display

See also

[getTextureRect](#), [setTexture](#)

void sf::Shape::update ()

Recompute the internal geometry of the shape.

This function must be called by the derived class everytime the shape's either `getPointCount` or `getPoint` is different).

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

- [RectangleShape.hpp](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Public Attributes](#) | [Static Public Attributes](#) | [List of all members](#)

sf::RenderStates Class Reference

[Graphics module](#)

Define the states used for drawing to a [RenderTarget](#). More...

```
#include <RenderStates.hpp>
```

Public Member Functions

`RenderStates ()`

Default constructor. [More...](#)

`RenderStates (const BlendMode &theBlendMode)`

Construct a default set of render states with a custom blend mode. [More...](#)

`RenderStates (const Transform &theTransform)`

Construct a default set of render states with a custom transform. [More...](#)

`RenderStates (const Texture *theTexture)`

Construct a default set of render states with a custom texture. [More...](#)

`RenderStates (const Shader *theShader)`

Construct a default set of render states with a custom shader. [More...](#)

`RenderStates (const BlendMode &theBlendMode, const Transform &`

`*theTexture, const Shader *theShader)`

Construct a set of render states with all its attributes. [More...](#)

Public Attributes

BlendMode `blendMode`

Blending mode. More...

Transform `transform`

Transform. More...

const Texture * `texture`

Texture. More...

const Shader * `shader`

Shader. More...

Static Public Attributes

static const `RenderStates` `Default`

Special instance holding the default render st

Detailed Description

Define the states used for drawing to a [RenderTarget](#).

There are four global states that can be applied to the drawn objects:

- the blend mode: how pixels of the object are blended with the background
- the transform: how the object is positioned/rotated/scaled
- the texture: what image is mapped to the object
- the shader: what custom effect is applied to the object

High-level objects such as sprites or text force some of these states when they will set its own texture, so that you don't have to care about it when

The transform is a special case: sprites, texts and shapes (and it's also true for all drawable classes too) combine their transform with the one that is passed to them. So that you can use a "global" transform on top of each object's transform.

Most objects, especially high-level drawables, can be drawn directly without specifying the render states – the default set of states is ok in most cases.

```
window.draw(sprite);
```

If you want to use a single specific render state, for example a shader, you can do this:

sf::RenderStates has an implicit one-argument constructor for each state.

```
window.draw(sprite, shader);
```

When you're inside the Draw function of a drawable object (inherited from [Drawable](#)), you can pass the render states unmodified, or change some of them. For example:

combine the current transform with its own transform. A sprite will set its transform to the result of this operation.

See also

[sf::RenderTarget](#), [sf::Drawable](#)

Definition at line 45 of file [RenderStates.hpp](#).

Constructor & Destructor Documentation

sf::RenderStates::RenderStates ()

Default constructor.

Constructing a default set of render states is equivalent to using `sf::RenderStates::operator=()`. This constructor also defines:

- the BlendAlpha blend mode
- the identity transform
- a null texture
- a null shader

sf::RenderStates::RenderStates (const BlendMode & theBlendMode)

Construct a default set of render states with a custom blend mode.

Parameters

theBlendMode Blend mode to use

sf::RenderStates::RenderStates (const Transform & theTransform)

Construct a default set of render states with a custom transform.

Parameters

theTransform Transform to use

sf::RenderStates::RenderStates (const Texture * theTexture)

Construct a default set of render states with a custom texture.

Parameters

theTexture Texture to use

sf::RenderStates::RenderStates (const Shader * theShader)

Construct a default set of render states with a custom shader.

Parameters

theShader Shader to use

sf::RenderStates::RenderStates (const BlendMode & theBlendMode, const Transform & theTransform, const Texture * theTexture, const Shader * theShader)

Construct a set of render states with all its attributes.

Parameters

theBlendMode Blend mode to use

theTransform Transform to use

theTexture Texture to use

theShader Shader to use

Member Data Documentation

BlendMode sf::RenderStates::blendMode

Blending mode.

Definition at line [115](#) of file RenderStates.hpp.

const RenderStates sf::RenderStates::Default

Special instance holding the default render states.

Definition at line [110](#) of file RenderStates.hpp.

const Shader* sf::RenderStates::shader

Shader.

Definition at line [118](#) of file RenderStates.hpp.

const Texture* sf::RenderStates::texture

Texture.

Definition at line [117](#) of file RenderStates.hpp.

Transform sf::RenderStates::transform

Transform.

Definition at line 116 of file RenderStates.hpp.

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

- [RenderStates.hpp](#)

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Classes](#) | [Public Member Functions](#) | [Protected Member Functions](#) | [List of all members](#)

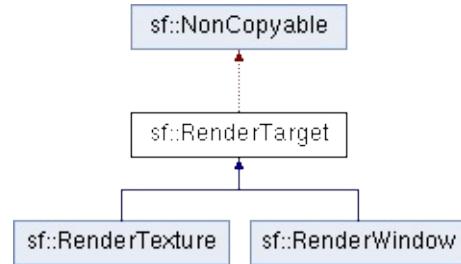
sf::RenderTarget Class Reference abstract

Graphics module

Base class for all render targets (window, texture, ...) [More...](#)

```
#include <RenderTarget.hpp>
```

Inheritance diagram for sf::RenderTarget:



Public Member Functions

virtual ~RenderTarget ()
Destructor. More...

void clear (const Color &color=Color(0, 0, 0, 255))
Clear the entire target with a single color. More...

void setView (const View &view)
Change the current active view. More...

const View & getView () const
Get the view currently in use in the render target. More...

const View & getDefaultView () const
Get the default view of the render target. More...

IntRect getViewport (const View &view) const
Get the viewport of a view, applied to this render target.

Vector2f mapPixelToCoords (const Vector2i &point) const
Convert a point from target coordinates to world coordinates.
More...

Vector2f mapPixelToCoords (const Vector2i &point, const View &view) const
Convert a point from target coordinates to world coordinates.

Vector2i mapCoordsToPixel (const Vector2f &point) const
Convert a point from world coordinates to target coordinates.
More...

Vector2i mapCoordsToPixel (const Vector2f &point, const View &view) const
Convert a point from world coordinates to target coordinates.

void `draw` (`const Drawable &drawable, const RenderStates`
Draw a drawable object to the render target. More...

void `draw` (`const Vertex *vertices, std::size_t vertexCount, F`
`RenderStates &states=RenderStates::Default)`
Draw primitives defined by an array of vertices. More...

virtual Vector2u `getSize () const =0`
Return the size of the rendering region of the target. More...

void `pushGLStates ()`
Save the current OpenGL render states and matrices. More...

void `popGLStates ()`
Restore the previously saved OpenGL render states an

void `resetGLStates ()`
Reset the internal OpenGL states so that the target is re

Protected Member Functions

`RenderTarget ()`

Default constructor. [More...](#)

`void initialize ()`

Performs the common initialization step after creation. [More...](#)

Detailed Description

Base class for all render targets (window, texture, ...)

`sf::RenderTarget` defines the common behavior of all the 2D render target

It makes it possible to draw 2D entities like sprites, shapes, text withc directly.

A `sf::RenderTarget` is also able to use views (`sf::View`), which are a kind can globally scroll, rotate or zoom everything that is drawn, without havin See the documentation of `sf::View` for more details and sample pieces of c

On top of that, render targets are still able to render direct OpenGL stuff. OpenGL calls and regular SFML drawing commands. When doing so, m not messed up by calling the `pushGLStates/popGLStates` functions.

See also

`sf::RenderWindow`, `sf::RenderTexture`, `sf::View`

Definition at line 51 of file `RenderTarget.hpp`.

Constructor & Destructor Documentation

virtual sf::RenderTarget::~RenderTarget()

Destructor.

sf::RenderTarget::RenderTarget()

Default constructor.

Member Function Documentation

```
void sf::RenderTarget::clear ( const Color & color = Color(0, 0, 0, 255) )
```

Clear the entire target with a single color.

This function is usually called once every frame, to clear the previous content.

Parameters

color Fill color to use to clear the render target

```
void sf::RenderTarget::draw ( const Drawable & drawable,
                            const RenderStates & states = RenderStates() )
```

Draw a drawable object to the render target.

Parameters

drawable Object to draw

states Render states to use for drawing

```
void sf::RenderTarget::draw ( const Vertex * vertices,
                            std::size_t vertexCount,
                            PrimitiveType type,
                            const RenderStates & states = RenderStates() )
```

Draw primitives defined by an array of vertices.

Parameters

vertices	Pointer to the vertices
vertexCount	Number of vertices in the array
type	Type of primitives to draw
states	Render states to use for drawing

`const View& sf::RenderTarget::getDefaultView() const`

Get the default view of the render target.

The default view has the initial size of the render target, and never changes after it is created.

Returns

The default view of the render target

See also

`setView`, `getView`

`virtual Vector2u sf::RenderTarget::getSize() const`

Return the size of the rendering region of the target.

Returns

Size in pixels

Implemented in `sf::RenderTexture`, and `sf::RenderWindow`.

const View& sf::RenderTarget::getView() const

Get the view currently in use in the render target.

Returns

The view object that is currently used

See also

[setView](#), [getDefaultView](#)

IntRect sf::RenderTarget::getViewport(const View & view) const

Get the viewport of a view, applied to this render target.

The viewport is defined in the view as a ratio, this function simply multiplies the dimensions of the render target to calculate the pixels rectangle that the view occupies in the target.

Parameters

view The view for which we want to compute the viewport

Returns

Viewport rectangle, expressed in pixels

void sf::RenderTarget::initialize()

Performs the common initialization step after creation.

The derived classes must call this function after the target is created and

Vector2i sf::RenderTarget::mapCoordsToPixel (const Vector2f & point)

Convert a point from world coordinates to target coordinates, using the current view.

This function is an overload of the mapCoordsToPixel function that implements the conversion equivalent to:

```
target.mapCoordsToPixel(point, target.getView());
```

Parameters

point Point to convert

Returns

The converted point, in target coordinates (pixels)

See also

[mapPixelToCoords](#)

Vector2i sf::RenderTarget::mapCoordsToPixel (const Vector2f & point, const View & view)

Convert a point from world coordinates to target coordinates.

This function finds the pixel of the render target that matches the given world coordinate. It does this by rendering the scene through the same process as the graphics card, to compute the final position of the point.

Initially, both coordinate systems (world units and target pixels) match. If you change the custom view or resize your render target, this assertion is not true anymore. For example, if your 2D world has a resolution of 75x50 pixels, a point at (10, 50) in your 2D world may map to the pixel (10, 50) of your render target, while a point at (140, 25).

This version uses a custom view for calculations, see the other overloads for more information. It also uses the current view of the render target.

Parameters

point Point to convert

view The view to use for converting the point

Returns

The converted point, in target coordinates (pixels)

See also

[mapPixelToCoords](#)

Vector2f sf::RenderTarget::mapPixelToCoords (const Vector2i & point, float x, float y)

Convert a point from target coordinates to world coordinates, using the current view.

This function is an overload of the mapPixelToCoords function that implements the conversion from target coordinates to world coordinates. It is equivalent to:

```
target.mapPixelToCoords(point, target.getView());
```

Parameters

point Pixel to convert

Returns

The converted point, in "world" coordinates

See also

[mapCoordsToPixel](#)

```
Vector2f sf::RenderTarget::mapPixelToCoords ( const Vector2i & point,
                                             const View & view,
                                             )
```

Convert a point from target coordinates to world coordinates.

This function finds the 2D position that matches the given pixel of the render target. It does the inverse of what the graphics card does, to find the initial position of the pixel.

Initially, both coordinate systems (world units and target pixels) match. If you change the custom view or resize your render target, this assertion is not true anymore. For example, the pixel at (50, 50) in your render target may map to the point (150, 75) in your 2D world space, or (140, 25).

For render-windows, this function is typically used to find which point the mouse cursor.

This version uses a custom view for calculations, see the other overloads if you want to use the current view of the render target.

Parameters

point Pixel to convert

view The view to use for converting the point

Returns

The converted point, in "world" units

See also

[mapCoordsToPixel](#)

```
void sf::RenderTarget::popGLStates ( )
```

Restore the previously saved OpenGL render states and matrices.

See the description of `pushGLStates` to get a detailed description of the

See also

[pushGLStates](#)

void sf::RenderTarget::pushGLStates()

Save the current OpenGL render states and matrices.

This function can be used when you mix SFML drawing and direct OpenGL code. It ensures that:

- SFML's internal states are not messed up by your OpenGL code
- your OpenGL states are not modified by a call to a SFML function

More specifically, it must be used around code that calls Draw functions.

```
// OpenGL code here...
window.pushGLStates();
window.draw(...);
window.draw(...);
window.popGLStates();
// OpenGL code here...
```

Note that this function is quite expensive: it saves all the possible OpenGL states you don't care about. Therefore it should be used wisely. It is probably best results will be achieved if you handle OpenGL states yourself (because they really changed, and need to be saved and restored). Take a look at the `sf::OpenGLState` class so.

See also

popGLStates

void sf::RenderTarget::resetGLStates()

Reset the internal OpenGL states so that the target is ready for drawing.

This function can be used when you mix SFML drawing and direct OpenGL code. It makes sure that all OpenGL state changes made by subsequent `draw()` calls will work as expected.

Example:

```
// OpenGL code here...
glPushAttrib(...);
window.resetGLStates();
window.draw(...);
window.draw(...);
glPopAttrib(...);
// OpenGL code here...
```

void sf::RenderTarget::setView(const View & view)

Change the current active view.

The view is like a 2D camera, it controls which part of the 2D scene is visible in the render target. The new view will affect everything that is drawn, until another view is set. The target keeps its own copy of the view object, so it is not necessary to call this function. To restore the original view of the target, you can pass `sf::View::None` to this function.

Parameters

view New view to use

See also

getView, getDefaultView

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

- [RenderTarget.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Member Functions](#) | [Protected Member Functions](#) | [List of all members](#)

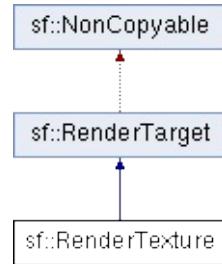
sf::RenderTexture Class Reference

Graphics module

Target for off-screen 2D rendering into a texture. More...

```
#include <RenderTexture.hpp>
```

Inheritance diagram for sf::RenderTexture:



Public Member Functions

`RenderTexture ()`
Default constructor. [More...](#)

`virtual ~RenderTexture ()`
Destructor. [More...](#)

`bool create (unsigned int width, unsigned int height, bool de)`
Create the render-texture. [More...](#)

`void setSmooth (bool smooth)`
Enable or disable texture smoothing. [More...](#)

`bool isSmooth () const`
Tell whether the smooth filtering is enabled or not. [More...](#)

`void setRepeated (bool repeated)`
Enable or disable texture repeating. [More...](#)

`bool isRepeated () const`
Tell whether the texture is repeated or not. [More...](#)

`bool generateMipmap ()`
Generate a mipmap using the current texture data. [More...](#)

`bool setActive (bool active=true)`
Activate or deactivate the render-texture for rendering.

`void display ()`
Update the contents of the target texture. [More...](#)

`virtual Vector2u getSize () const`

Return the size of the rendering region of the texture. More...

const Texture & **getTexture () const**
Get a read-only reference to the target texture. More...

void **clear (const Color &color=Color(0, 0, 0, 255))**
Clear the entire target with a single color. More...

void **setView (const View &view)**
Change the current active view. More...

const View & **getView () const**
Get the view currently in use in the render target. More...

const View & **getDefaultView () const**
Get the default view of the render target. More...

IntRect **getViewport (const View &view) const**
Get the viewport of a view, applied to this render target.

Vector2f **mapPixelToCoords (const Vector2i &point) const**
Convert a point from target coordinates to world coordinates.
More...

Vector2f **mapPixelToCoords (const Vector2i &point, const View &view) const**
Convert a point from target coordinates to world coordinates.

Vector2i **mapCoordsToPixel (const Vector2f &point) const**
Convert a point from world coordinates to target coordinates.
More...

Vector2i **mapCoordsToPixel (const Vector2f &point, const View &view) const**
Convert a point from world coordinates to target coordinates.

void **draw (const Drawable &drawable, const RenderStates &states) const**

Draw a drawable object to the render target. [More...](#)

`void draw (const Vertex *vertices, std::size_t vertexCount, F
RenderStates &states=RenderStates::Default)`
Draw primitives defined by an array of vertices. [More...](#)

`void pushGLStates ()`
Save the current OpenGL render states and matrices. [More...](#)

`void popGLStates ()`
Restore the previously saved OpenGL render states an

`void resetGLStates ()`
Reset the internal OpenGL states so that the target is re

Protected Member Functions

void `initialize ()`

Performs the common initialization step after creation. More...

Detailed Description

Target for off-screen 2D rendering into a texture.

`sf::RenderTexture` is the little brother of `sf::RenderWindow`.

It implements the same 2D drawing and OpenGL-related functions (see for more details), the difference is that the result is stored in an off-screen a window.

Rendering to a texture can be useful in a variety of situations:

- precomputing a complex static texture (like a level's background from i
- applying post-effects to the whole scene with shaders
- creating a sprite from a 3D object rendered with OpenGL
- etc.

Usage example:

```
// Create a new render-window
sf::RenderWindow window(sf::VideoMode(800, 600), "SFML window");

// Create a new render-texture
sf::RenderTexture texture;
if (!texture.create(500, 500))
    return -1;

// The main loop
while (window.isOpen())
{
    // Event processing
    // ...

    // Clear the whole texture with red color
    texture.clear(sf::Color::Red);

    // Draw stuff to the texture
    texture.draw(sprite); // sprite is a sf::Sprite
```

```
    texture.draw(shape);      // shape is a sf::Shape
    texture.draw(text);       // text is a sf::Text

    // We're done drawing to the texture
    texture.display();

    // Now we start rendering to the window, clear it first
    window.clear();

    // Draw the texture
    sf::Sprite sprite(texture.getTexture());
    window.draw(sprite);

    // End the current frame and display its contents on screen
    window.display();
}
```

Like `sf::RenderTarget`, `sf::RenderTexture` is still able to render direct OpenGL calls. You can mix together OpenGL calls and regular SFML drawing commands. If you're doing OpenGL rendering, don't forget to request it when calling `RenderTexture::create`.

See also

`sf::RenderTarget`, `sf::RenderWindow`, `sf::View`, `sf::Texture`

Definition at line 47 of file `RenderTexture.hpp`.

Constructor & Destructor Documentation

sf::RenderTexture::RenderTexture ()

Default constructor.

Constructs an empty, invalid render-texture. You must call `create` to have

See also

`create`

virtual sf::RenderTexture::~RenderTexture ()

Destructor.

Member Function Documentation

```
void sf::RenderTarget::clear ( const Color & color = Color(0, 0, 0, 255) )
```

Clear the entire target with a single color.

This function is usually called once every frame, to clear the previous color.

Parameters

color Fill color to use to clear the render target

```
bool sf::RenderTarget::create ( unsigned int width,
                               unsigned int height,
                               bool           depthBuffer = false
                           )
```

Create the render-texture.

Before calling this function, the render-texture is in an invalid state, thus doing anything with the render-texture. The last parameter, *depthBuffer* creates a render-texture for 3D OpenGL rendering that requires a depth buffer. Consider you should leave this parameter to false (which is its default value).

Parameters

width Width of the render-texture

height Height of the render-texture

depthBuffer Do you want this render-texture to have a depth buffer

Returns

True if creation has been successful

`void sf::RenderTexture::display()`

Update the contents of the target texture.

This function updates the target texture with what has been drawn so far. This function is mandatory at the end of rendering. Not calling it may leave the texture empty.

`void sf::RenderTarget::draw(const Drawable & drawable, const RenderStates & states = RenderStates())`

Draw a drawable object to the render target.

Parameters

drawable Object to draw
states Render states to use for drawing

`void sf::RenderTarget::draw(const Vertex * vertices, std::size_t vertexCount, PrimitiveType type, const RenderStates & states = RenderStates())`

Draw primitives defined by an array of vertices.

Parameters

vertices	Pointer to the vertices
vertexCount	Number of vertices in the array
type	Type of primitives to draw
states	Render states to use for drawing

bool sf::RenderTexture::generateMipmap ()

Generate a mipmap using the current texture data.

This function is similar to [Texture::generateMipmap](#) and operates on the drawing. Be aware that any draw operation may modify the base level. Calling this function only makes sense after all drawing is completed as calling [display](#) after subsequent drawing will lead to undefined behavior generated.

Returns

True if mipmap generation was successful, false if unsuccessful

const View& sf::RenderTarget::getDefaultView () const

Get the default view of the render target.

The default view has the initial size of the render target, and never changes once created.

Returns

The default view of the render target

See also

[setView](#), [getView](#)

virtual Vector2u sf::RenderTexture::getSize() const

Return the size of the rendering region of the texture.

The returned value is the size that you passed to the create function.

Returns

Size in pixels

Implements [sf::RenderTarget](#).

const Texture& sf::RenderTexture::getTexture() const

Get a read-only reference to the target texture.

After drawing to the render-texture and calling [Display](#), you can retrieve the texture function, and draw it using a sprite (for example). The internal [sf::Texture](#) is shared by all targets, so that it is possible to call this function once and even after it is modified.

Returns

Const reference to the texture

const View& sf::RenderTarget::getView() const

Get the view currently in use in the render target.

Returns

The view object that is currently used

See also

[setView](#), [getDefaultView](#)

IntRect sf::RenderTarget::getViewport (const View & view) const

Get the viewport of a view, applied to this render target.

The viewport is defined in the view as a ratio, this function simply multiplies the dimensions of the render target to calculate the pixels rectangle that the view occupies in the target.

Parameters

view The view for which we want to compute the viewport

Returns

Viewport rectangle, expressed in pixels

void sf::RenderTarget::initialize ()

Performs the common initialization step after creation.

The derived classes must call this function after the target is created and before rendering.

bool sf::RenderTarget::isRepeated () const

Tell whether the texture is repeated or not.

Returns

True if texture is repeated

See also

[setRepeated](#)

bool sf::RenderTexture::isSmooth() const

Tell whether the smooth filtering is enabled or not.

Returns

True if texture smoothing is enabled

See also

[setSmooth](#)

Vector2i sf::RenderTarget::mapCoordsToPixel(const Vector2f & point) const

Convert a point from world coordinates to target coordinates, using the current view.

This function is an overload of the `mapCoordsToPixel` function that implements the conversion from world coordinates to target coordinates. It is equivalent to:

```
target.mapCoordsToPixel(point, target.getView());
```

Parameters

point Point to convert

Returns

The converted point, in target coordinates (pixels)

See also

[mapPixelToCoords](#)

```
Vector2i sf::RenderTarget::mapCoordsToPixel ( const Vector2f & point,
                                             const View & view,
                                             )
```

Convert a point from world coordinates to target coordinates.

This function finds the pixel of the render target that matches the given point through the same process as the graphics card, to compute the final position.

Initially, both coordinate systems (world units and target pixels) match. If you change the custom view or resize your render target, this assertion is not true anymore. For example, the point (75, 75) in your 2D world may map to the pixel (10, 50) of your render target, while the point (140, 25) maps to (10, 50).

This version uses a custom view for calculations, see the other overloads if you want to use the current view of the render target.

Parameters

point Point to convert

view The view to use for converting the point

Returns

The converted point, in target coordinates (pixels)

See also

[mapPixelToCoords](#)

```
Vector2f sf::RenderTarget::mapPixelToCoords ( const Vector2i & point,
```

Convert a point from target coordinates to world coordinates, using the current view of the render target.

This function is an overload of the `mapPixelToCoords` function that implements the inverse of what the graphics card does, to find the initial position of a pixel.

```
target.mapPixelToCoords(point, target.getView());
```

Parameters

point Pixel to convert

Returns

The converted point, in "world" coordinates

See also

[mapCoordsToPixel](#)

Vector2f sf::RenderTarget::mapPixelToCoords (const Vector2i & point, const View & view, const RenderWindow & window)

Convert a point from target coordinates to world coordinates.

This function finds the 2D position that matches the given pixel of the render target. It does the inverse of what the graphics card does, to find the initial position of a pixel.

Initially, both coordinate systems (world units and target pixels) match. If you change the custom view or resize your render target, this assertion is not true anymore. For example, if you have a render target of size 200x150 and a custom view of size 150x100, a pixel at (50, 75) in your render target may map to the point (150, 75) in your 2D world space, but to (140, 25) in your 150x100 view.

For render-windows, this function is typically used to find which point underlies the mouse cursor.

This version uses a custom view for calculations, see the other overloads for more details.

use the current view of the render target.

Parameters

point Pixel to convert

view The view to use for converting the point

Returns

The converted point, in "world" units

See also

[mapCoordsToPixel](#)

void sf::RenderTarget::popGLStates ()

Restore the previously saved OpenGL render states and matrices.

See the description of [pushGLStates](#) to get a detailed description of the

See also

[pushGLStates](#)

void sf::RenderTarget::pushGLStates ()

Save the current OpenGL render states and matrices.

This function can be used when you mix SFML drawing and direct OpenGL rendering. When you call [popGLStates](#), it ensures that:

- SFML's internal states are not messed up by your OpenGL code
- your OpenGL states are not modified by a call to a SFML function

More specifically, it must be used around code that calls Draw functions.

```
// OpenGL code here...
window.pushGLStates();
window.draw(...);
window.draw(...);
window.popGLStates();
// OpenGL code here...
```

Note that this function is quite expensive: it saves all the possible OpenGL states you don't care about. Therefore it should be used wisely. It is probably best results will be achieved if you handle OpenGL states yourself (because they really changed, and need to be saved and restored). Take a look at the [so](#).

See also

[popGLStates](#)

void sf::RenderTarget::resetGLStates ()

Reset the internal OpenGL states so that the target is ready for drawing.

This function can be used when you mix SFML drawing and direct OpenGL drawing to use pushGLStates/popGLStates. It makes sure that all OpenGL state changes made by OpenGL code between subsequent [draw\(\)](#) calls will work as expected.

Example:

```
// OpenGL code here...
glPushAttrib(...);
window.resetGLStates();
window.draw(...);
window.draw(...);
glPopAttrib(...);
// OpenGL code here...
```

bool sf::RenderTexture:: setActive (bool active = true)

Activate or deactivate the render-texture for rendering.

This function makes the render-texture's context current for future OpenGL operations (OpenGL won't care about it if you're not doing direct OpenGL stuff). Only one thread can have a render-target active at a time. If you switch threads, so if you want to draw OpenGL geometry to another render target, you must forget to activate it again.

Parameters

active True to activate, false to deactivate

Returns

True if operation was successful, false otherwise

void sf::RenderTexture:: setRepeated (bool repeated)

Enable or disable texture repeating.

This function is similar to [Texture::setRepeated](#). This parameter is disabled by default.

Parameters

repeated True to enable repeating, false to disable it

See also

[isRepeated](#)

void sf::RenderTexture:: setSmooth (bool smooth)

Enable or disable texture smoothing.

This function is similar to `Texture::setSmooth`. This parameter is disablec

Parameters

smooth True to enable smoothing, false to disable it

See also

`isSmooth`

void sf::RenderTarget::setView (const View & view)

Change the current active view.

The view is like a 2D camera, it controls which part of the 2D scene is vi
render target. The new view will affect everything that is drawn, until
target keeps its own copy of the view object, so it is not necessary to
calling this function. To restore the original view of the target, you can pa
to this function.

Parameters

view New view to use

See also

`getView`, `getDefaultView`

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

- `RenderTexture.hpp`

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Member Functions](#) | [Protected Member Functions](#) | [List of all members](#)

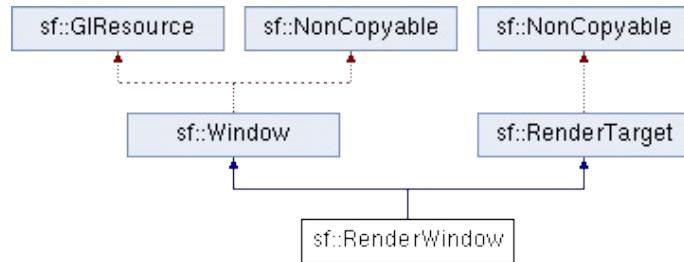
sf::RenderWindow Class Reference

Graphics module

Window that can serve as a target for 2D drawing. [More...](#)

```
#include <RenderWindow.hpp>
```

Inheritance diagram for sf::RenderWindow:



Public Member Functions

`RenderWindow ()`

Default constructor. [More...](#)

`RenderWindow (VideoMode mode, const String &title, const ContextSettings &settings=ContextSettings())`

Construct a new window. [More...](#)

`RenderWindow (WindowHandle handle, const ContextSettings &settings=ContextSettings())`

Construct the window from an existing control.

`virtual ~RenderWindow ()`

Destructor. [More...](#)

`virtual Vector2u getSize () const`

Get the size of the rendering region of the window.

`Image capture () const`

Copy the current contents of the window to an image.

`void create (VideoMode mode, const String &title, const ContextSettings &settings=ContextSettings())`

Create (or recreate) the window. [More...](#)

`void create (WindowHandle handle, const ContextSettings &settings=ContextSettings())`

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

`void close ()`

Close the window and destroy all the attached contexts.

`bool isOpen () const`
Tell whether or not the window is open. [More...](#)

`const ContextSettings & getSettings () const`
Get the settings of the OpenGL context of the window. [More...](#)

`bool pollEvent (Event &event)`
Pop the event on top of the event queue, if any.

`bool waitEvent (Event &event)`
Wait for an event and return it. [More...](#)

`Vector2i getPosition () const`
Get the position of the window. [More...](#)

`void setPosition (const Vector2i &position)`
Change the position of the window on screen. [More...](#)

`void setSize (const Vector2u &size)`
Change the size of the rendering region of the window. [More...](#)

`void setTitle (const String &title)`
Change the title of the window. [More...](#)

`void setIcon (unsigned int width, unsigned int height)`
Change the window's icon. [More...](#)

`void setVisible (bool visible)`
Show or hide the window. [More...](#)

`void setVerticalSyncEnabled (bool enabled)`
Enable or disable vertical synchronization. [More...](#)

`void setMouseCursorVisible (bool visible)`
Show or hide the mouse cursor. [More...](#)

void [setMouseCursorGrabbed](#) (bool grabbed)
Grab or release the mouse cursor. [More...](#)

void [setKeyRepeatEnabled](#) (bool enabled)
Enable or disable automatic key-repeat. [More..](#)

void [setFramerateLimit](#) (unsigned int limit)
Limit the framerate to a maximum fixed frequer

void [setJoystickThreshold](#) (float threshold)
Change the joystick threshold. [More...](#)

bool [setActive](#) (bool active=true) const
Activate or deactivate the window as the currer
[More...](#)

void [requestFocus](#) ()
Request the current window to be made the ac

bool [hasFocus](#) () const
Check whether the window has the input focus

void [display](#) ()
Display on screen what has been rendered to t

WindowHandle [getSystemHandle](#) () const
Get the OS-specific handle of the window. [Mor](#)

void [clear](#) (const [Color](#) &color=[Color](#)(0, 0, 0, 255))
Clear the entire target with a single color. [More](#)

void [setView](#) (const [View](#) &view)
Change the current active view. [More...](#)

`const View & getView () const`
Get the view currently in use in the render target.

`const View & getDefaultView () const`
Get the default view of the render target. More.

`IntRect getViewport (const View &view) const`
Get the viewport of a view, applied to this render target.

`Vector2f mapPixelToCoords (const Vector2i &point) const`
Convert a point from target coordinates to world coordinates of a view. More...

`Vector2f mapPixelToCoords (const Vector2i &point, const View &view) const`
Convert a point from target coordinates to world coordinates of a view.

`Vector2i mapCoordsToPixel (const Vector2f &point) const`
Convert a point from world coordinates to target coordinates of a view. More...

`Vector2i mapCoordsToPixel (const Vector2f &point, const View &view) const`
Convert a point from world coordinates to target coordinates of a view.

`void draw (const Drawable &drawable, const RenderStates &states=RenderStates::Default)`
Draw a drawable object to the render target. More.

`void draw (const Vertex *vertices, std::size_t vertexCount, const RenderStates &states=RenderStates::Default)`
Draw primitives defined by an array of vertices.

`void pushGLStates ()`
Save the current OpenGL render states and make them available for restoration.

`void popGLStates ()`
Restore the previously saved OpenGL render states.

```
void resetGLStates ()
```

Reset the internal OpenGL states so that the ta
More...

Protected Member Functions

`virtual void onCreate ()`

Function called after the window has been created. [More...](#)

`virtual void onResize ()`

Function called after the window has been resized. [More...](#)

`void initialize ()`

Performs the common initialization step after creation. [More.](#)

Detailed Description

Window that can serve as a target for 2D drawing.

`sf::RenderWindow` is the main class of the Graphics module.

It defines an OS window that can be painted using the other classes of the Graphics module.

`sf::RenderWindow` is derived from `sf::Window`, thus it inherits all its features (such as OpenGL rendering, etc. See the documentation of `sf::Window` for a more detailed description). It also adds some specific features related to 2D drawing (such as its base class `sf::RenderTarget` for more details). Here is a typical rendering loop using `sf::RenderWindow`:

```
// Declare and create a new render-window
sf::RenderWindow window(sf::VideoMode(800, 600), "SFML window");

// Limit the framerate to 60 frames per second (this step is optional)
window.setFramerateLimit(60);

// The main loop - ends as soon as the window is closed
while (window.isOpen())
{
    // Event processing
    sf::Event event;
    while (window.pollEvent(event))
    {
        // Request for closing the window
        if (event.type == sf::Event::Closed)
            window.close();
    }

    // Clear the whole window before rendering a new frame
    window.clear();

    // Draw some graphical entities
    window.draw(sprite);
    window.draw(circle);
```

```
    window.draw(text);

    // End the current frame and display its contents on screen
    window.display();
}
```

Like `sf::Window`, `sf::RenderWindow` is still able to render direct OpenGL together OpenGL calls and regular SFML drawing commands.

```
// Create the render window
sf::RenderWindow window(sf::VideoMode(800, 600), "SFML OpenGL");

// Create a sprite and a text to display
sf::Sprite sprite;
sf::Text text;
...

// Perform OpenGL initializations
glMatrixMode(GL_PROJECTION);
...

// Start the rendering loop
while (window.isOpen())
{
    // Process events
    ...

    // Draw a background sprite
    window.pushGLStates();
    window.draw(sprite);
    window.popGLStates();

    // Draw a 3D object using OpenGL
    glBegin(GL_QUADS);
        glVertex3f(...);
    ...
    glEnd();

    // Draw text on top of the 3D object
    window.pushGLStates();
    window.draw(text);
    window.popGLStates();

    // Finally, display the rendered frame on screen
    window.display();
}
```

See also

`sf::Window`, `sf::RenderTarget`, `sf::RenderTexture`, `sf::View`

Definition at line 44 of file `RenderWindow.hpp`.

Constructor & Destructor Documentation

`sf::RenderWindow::RenderWindow()`

Default constructor.

This constructor doesn't actually create the window, use the other constr

```
sf::RenderWindow::RenderWindow( VideoMode mode,
                               const String & title,
                               Uint32 style,
                               const ContextSettings & settings)
)
```

Construct a new window.

This constructor creates the window with the size and pixel depth defined by the first two parameters. The third parameter is an optional string to be passed to customize the look and behavior of the window (borders, tit

The fourth parameter is an optional structure specifying advanced OpenGL settings like antialiasing, depth-buffer bits, etc. You shouldn't care about these parameters if you're not using the graphics module.

Parameters

mode Video mode to use (defines the width, height and depth of the window)

title Title of the window

style Window style, a bitwise OR combination of `sf::Style` enum values

settings Additional settings for the underlying OpenGL context

```
sf::RenderWindow::RenderWindow( WindowHandle handle  
                               const ContextSettings & settings  
)
```

Construct the window from an existing control.

Use this constructor if you want to create an SFML rendering area into an existing window.

The second parameter is an optional structure specifying advanced OpenGL settings such as antialiasing, depth-buffer bits, etc. You shouldn't care about these parameters if you're not familiar with the graphics module.

Parameters

handle Platform-specific handle of the control (*HWND* on Windows, *NSWindow* on OS X)

settings Additional settings for the underlying OpenGL context

```
virtual sf::RenderWindow::~RenderWindow( )
```

Destructor.

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

Member Function Documentation

Image sf::RenderWindow::capture () const

Copy the current contents of the window to an image.

Deprecated:

Use a `sf::Texture` and its `sf::Texture::update(const Window&)` function instead.

```
sf::Vector2u windowHeight = window.getSize();
sf::Texture texture;
texture.create(windowSize.x, windowHeight.y);
texture.update(window);
sf::Image screenshot = texture.copyToImage();
```

This is a slow operation, whose main purpose is to make screenshots. Instead, update an image with the contents of the window and then use it for drawing. Use a `sf::Texture` and its `update(Window&)` function. You can also draw things directly on a `sf::RenderTarget`.

Returns

`Image` containing the captured contents

void sf::RenderTarget::clear (const Color & color = color(0, 0, 0, 255))

Clear the entire target with a single color.

This function is usually called once every frame, to clear the previous color.

Parameters

color Fill color to use to clear the render target

void sf::Window::close()

Close the window and destroy all the attached resources.

After calling this function, the `sf::Window` instance remains valid and you can still interact with your window. All other functions such as `pollEvent()` or `display()` will still work (they will simply ignore the window if it is closed). The `isOpen()` function will return `false` (unless you call `open()` again), and will have no effect on closed windows.

```
void sf::Window::create ( VideoMode mode,  
                         const String & title,  
                         Uint32 style = Style::Default,  
                         const ContextSettings & settings = ContextSettings(),  
                         const ContextSettings & settingsForOpenGL = ContextSettings());
```

Create (or recreate) the window.

If the window was already created, it closes it first. If *style* contains Style a valid video mode.

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

Parameters

mode Video mode to use (defines the width, height and depth of window)

title Title of the window

style Window style, a bitwise OR combination of sf::Style enum

settings Additional settings for the underlying OpenGL context

```
void sf::Window::create ( WindowHandle handle,
const ContextSettings & settings = contextSettings )
```

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

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

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

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 to show it on screen.

```
void sf::RenderTarget::draw ( const Drawable & drawable,
const RenderStates & states = RenderStates() )
```

Draw a drawable object to the render target.

Parameters

- drawable** Object to draw
states Render states to use for drawing

```
void sf::RenderTarget::draw ( const Vertex *  
                           std::size_t  
                           PrimitiveType  
                           const RenderStates & states = RenderS  
                           )
```

Draw primitives defined by an array of vertices.

Parameters

- vertices** Pointer to the vertices
vertexCount Number of vertices in the array
type Type of primitives to draw
states Render states to use for drawing

```
const View& sf::RenderTarget::getDefaultView ( ) const
```

Get the default view of the render target.

The default view has the initial size of the render target, and never changes after it is created.

Returns

The default view of the render target

See also

[setView](#), [getView](#)

Vector2i sf::Window::getPosition() const

Get the position of the window.

Returns

Position of the window, in pixels

See also

[setPosition](#)

const ContextSettings& 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 function, if one or more settings were not supported. In this case, SFML creates a new structure with the supported settings.

Returns

Structure containing the OpenGL context settings

virtual Vector2u sf::RenderWindow::getSize() const

Get the size of the rendering region of the window.

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

Returns

Size in pixels

Implements [sf::RenderTarget](#).

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 typed pointer to the OS. You shouldn't need to use this function, unless you have very specific needs. SFML doesn't support, or implement a temporary workaround until a bug is fixed.

Returns

System handle of the window

const View& sf::RenderTarget::getView() const

Get the view currently in use in the render target.

Returns

The view object that is currently used

See also

[setView](#), [getDefaultView](#)

IntRect sf::RenderTarget::getViewport(const View & view) const

Get the viewport of a view, applied to this render target.

The viewport is defined in the view as a ratio, this function simply multiplies it by the dimensions of the render target to calculate the pixels rectangle that the view covers in the target.

Parameters

view The view for which we want to compute the viewport

Returns

Viewport rectangle, expressed in pixels

bool sf::Window::hasFocus() const

Check whether the window has the input focus.

At any given time, only one window may have the input focus to receive or most mouse events.

Returns

True if window has focus, false otherwise

See also

[requestFocus](#)

void sf::RenderTarget::initialize()

Performs the common initialization step after creation.

The derived classes must call this function after the target is created and

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 is considered open (therefore this function would return true).

Returns

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

Vector2i sf::RenderTarget::mapCoordsToPixel (const Vector2f & point, View & view)

Convert a point from world coordinates to target coordinates, using the current view.

This function is an overload of the mapCoordsToPixel function that implements the conversion equivalent to:

```
target.mapCoordsToPixel(point, target.getView());
```

Parameters

point Point to convert

Returns

The converted point, in target coordinates (pixels)

See also

[mapPixelToCoords](#)

Vector2i sf::RenderTarget::mapCoordsToPixel (const Vector2f & point, View & view, const Color & color)

Convert a point from world coordinates to target coordinates.

This function finds the pixel of the render target that matches the given

through the same process as the graphics card, to compute the final position.

Initially, both coordinate systems (world units and target pixels) match. When you create a custom view or resize your render target, this assertion is not true anymore. For example, if you have a point at (75, 75) in your 2D world, it may map to the pixel (10, 50) of your render target, and then to (140, 25).

This version uses a custom view for calculations, see the other overloads if you want to use the current view of the render target.

Parameters

point Point to convert

view The view to use for converting the point

Returns

The converted point, in target coordinates (pixels)

See also

[mapPixelToCoords](#)

Vector2f sf::RenderTarget::mapPixelToCoords (const Vector2i & point)

Convert a point from target coordinates to world coordinates, using the current view.

This function is an overload of the `mapPixelToCoords` function that implements the conversion using the current view. It is equivalent to:

```
target.mapPixelToCoords(point, target.getView());
```

Parameters

point Pixel to convert

Returns

The converted point, in "world" coordinates

See also

[mapCoordsToPixel](#)

Vector2f sf::RenderTarget::mapPixelToCoords (const Vector2i & point, const View & view)

Convert a point from target coordinates to world coordinates.

This function finds the 2D position that matches the given pixel of the render target. It does the inverse of what the graphics card does, to find the initial position of the mouse cursor.

Initially, both coordinate systems (world units and target pixels) match. If you change your custom view or resize your render target, this assertion is not true anymore. For example, if you have a render target of size 50x50 and a window of size 150x75, a pixel at (50, 50) in your render target may map to the point (150, 75) in your 2D window, but to (140, 25) in your 2D world.

For render-windows, this function is typically used to find which point corresponds to the current position of the mouse cursor.

This version uses a custom view for calculations, see the other overloads for more information. If you do not provide a view, it will use the current view of the render target.

Parameters

point Pixel to convert

view The view to use for converting the point

Returns

The converted point, in "world" units

See also

mapCoordsToPixel

virtual void sf::RenderWindow::onCreate ()

Function called after the window has been created.

This function is called so that derived classes can perform their own specific actions after the window is created.

Reimplemented from `sf::Window`.

virtual void sf::RenderWindow::onResize ()

Function called after the window has been resized.

This function is called so that derived classes can perform custom actions after the window has been resized.

Reimplemented from `sf::Window`.

bool sf::Window::pollEvent (`Event & 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 the event parameter will remain unmodified. Note that more than one event may be present in the event queue at once. You must call this function in a loop to make sure that you process every pending event.

```
sf::Event event;
while (window.pollEvent(event))
```

```
{  
    // process event...  
}
```

Parameters

event [Event](#) to be returned

Returns

True if an event was returned, or false if the event queue was empty

See also

[waitEvent](#)

void sf::RenderTarget::popGLStates ()

Restore the previously saved OpenGL render states and matrices.

See the description of [pushGLStates](#) to get a detailed description of the

See also

[pushGLStates](#)

void sf::RenderTarget::pushGLStates ()

Save the current OpenGL render states and matrices.

This function can be used when you mix SFML drawing and direct OpenGL drawing. It ensures that:

- SFML's internal states are not messed up by your OpenGL code
- your OpenGL states are not modified by a call to a SFML function

More specifically, it must be used around code that calls Draw functions.

```
// OpenGL code here...
window.pushGLStates();
window.draw(...);
window.draw(...);
window.popGLStates();
// OpenGL code here...
```

Note that this function is quite expensive: it saves all the possible OpenGL states you don't care about. Therefore it should be used wisely. It is probably best results will be achieved if you handle OpenGL states yourself (because they really changed, and need to be saved and restored). Take a look at the [sf::OpenGLState](#) class so.

See also

[popGLStates](#)

void sf::Window::requestFocus()

Request the current window to be made the active foreground window.

At any given time, only one window may have the input focus to receive keyboard or mouse events. If a window requests focus, it only hints to the operating system that it wants to be focused. The operating system is free to deny the request. This is not to say that the window will never receive input events.

See also

[hasFocus](#)

void sf::RenderTarget::resetGLStates()

Reset the internal OpenGL states so that the target is ready for drawing.

This function can be used when you mix SFML drawing and direct OpenGL code to use pushGLStates/popGLStates. It makes sure that all OpenGL states set by subsequent `draw()` calls will work as expected.

Example:

```
// OpenGL code here...
glPushAttrib(...);
window.resetGLStates();
window.draw(...);
window.draw(...);
glPopAttrib(...);
// OpenGL code here...
```

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, deactivate it on the previous thread first if it was active. Only one window can be active at a time, thus the window previously active (if any) automatically gets deactivated when you activate another window with `requestFocus()`.

Parameters

active True to activate, false to deactivate

Returns

True 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 `pollEvent()` to ensure the framerate is not exceeded.

frame lasted long enough to match the framerate limit. SFML will try to reach it can, but since it internally uses sf::sleep, whose precision depends on the system, it may be a little unprecise as well (for example, you can get 65 FPS when using this function).

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. The pixels are copied from the source alive after calling this function.

See also

[setTitle](#)

```
void sf::Window::setJoystickThreshold ( float threshold )
```

Change the joystick threshold.

The joystick threshold is the value below which no JoystickMoved event

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 when the key is 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::setMouseCursorGrabbed (bool grabbed)

Grab or release the mouse cursor.

If set, grabs the mouse cursor inside this window's client area so it may not leave the window's bounds. Note that grabbing is only active while the window has focus.

Parameters

grabbed 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 & 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 of a child window/control).

Parameters

position New position, in pixels

See also

[getPosition](#)

void sf::Window::setSize (const Vector2u & size)

Change the size of the rendering region of the window.

Parameters

size New size, in pixels

See also

[getSize](#)

void sf::Window::setTitle (const String & title)

Change the title of the window.

Parameters

title New title

See also

[setIcon](#)

void sf::Window::setVerticalSyncEnabled (bool enabled)

Enable or disable vertical synchronization.

Activating vertical synchronization will limit the number of frames displayed on the monitor. This can avoid some visual artifacts, and limit the framerate to the same across different computers).

Vertical synchronization is disabled by default.

Parameters

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

void sf::RenderTarget::setView (const View & view)

Change the current active view.

The view is like a 2D camera, it controls which part of the 2D scene is visible on the render target. The new view will affect everything that is drawn, until another view is set.

target keeps its own copy of the view object, so it is not necessary to calling this function. To restore the original view of the target, you can pa to this function.

Parameters

view New view to use

See also

[getView](#), [getDefaultView](#)

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 & event)

Wait for an event and return it.

This function is blocking: if there's no pending event then it will wait unt function returns (and no error occurred), the *event* object is always valid is typically used when you have a thread that is dedicated to events l thread sleep as long as no new event is received.

```
sf::Event event;
if (window.waitEvent(event))
{
    // process event...
```

```
}
```

Parameters

event Event to be returned

Returns

False if any error occurred

See also

[pollEvent](#)

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

- [RenderWindow.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Classes](#) | [Public Types](#) | [Public Member Functions](#) | [Static Public Member Functions](#) | [Static Public Attributes](#) | [Static Private Attributes](#)

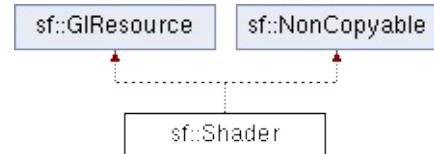
sf::Shader Class Reference

Graphics module

Shader class (vertex, geometry and fragment) [More...](#)

```
#include <Shader.hpp>
```

Inheritance diagram for sf::Shader:



Classes

struct `CurrentTextureType`

Special type that can be passed to `setUniform()`, and that represents the current texture being drawn. [More...](#)

Public Types

```
enum Type { Vertex, Geometry, Fragment }
```

Types of shaders. More...

Public Member Functions

`Shader ()`

Default constructor. [More...](#)

`~Shader ()`

Destructor. [More...](#)

`bool loadFromFile (const std::string &filename, Type type)`

Load the vertex, geometry or fragment shader from a file. [More...](#)

`bool loadFromFile (const std::string &vertexShaderFilename, const std::string &fragmentShaderFilename)`

Load both the vertex and fragment shaders from files. [More...](#)

`bool loadFromFile (const std::string &vertexShaderFilename, const std::string &geometryShaderFilename, const std::string &fragmentShaderFilename)`

Load the vertex, geometry and fragment shaders from files

`bool loadFromMemory (const std::string &shader, Type type)`

Load the vertex, geometry or fragment shader from a source code string.

`bool loadFromMemory (const std::string &vertexShader, const std::string &fragmentShader)`

Load both the vertex and fragment shaders from source code strings.

`bool loadFromMemory (const std::string &vertexShader, const std::string &geometryShader, const std::string &fragmentShader)`

Load the vertex, geometry and fragment shaders from source code strings.

`bool loadFromStream (InputStream &stream, Type type)`

Load the vertex, geometry or fragment shader from a custom stream.

`bool loadFromStream (InputStream &vertexShaderStream, InputStream &geometryShaderStream, InputStream &fragmentShaderStream)`

`&fragmentShaderStream)`

Load both the vertex and fragment shaders from custom st

`bool loadFromStream (InputStream &vertexShaderStream, Inpu`
`&geometryShaderStream, InputStream &fragmentShaderS`
Load the vertex, geometry and fragment shaders from cust

`void setUniform (const std::string &name, float x)`
Specify value for `float` uniform. [More...](#)

`void setUniform (const std::string &name, const GIsL::Vec2 &ve`
Specify value for `vec2` uniform. [More...](#)

`void setUniform (const std::string &name, const GIsL::Vec3 &ve`
Specify value for `vec3` uniform. [More...](#)

`void setUniform (const std::string &name, const GIsL::Vec4 &ve`
Specify value for `vec4` uniform. [More...](#)

`void setUniform (const std::string &name, int x)`
Specify value for `int` uniform. [More...](#)

`void setUniform (const std::string &name, const GIsL::Ivec2 &ve`
Specify value for `ivec2` uniform. [More...](#)

`void setUniform (const std::string &name, const GIsL::Ivec3 &ve`
Specify value for `ivec3` uniform. [More...](#)

`void setUniform (const std::string &name, const GIsL::Ivec4 &ve`
Specify value for `ivec4` uniform. [More...](#)

`void setUniform (const std::string &name, bool x)`
Specify value for `bool` uniform. [More...](#)

`void setUniform (const std::string &name, const GIsL::Bvec2 &ve`

Specify value for `bvec2` uniform. [More...](#)

void `setUniform` (const std::string &name, const `Gsl::Bvec3` &value)
Specify value for `bvec3` uniform. [More...](#)

void `setUniform` (const std::string &name, const `Gsl::Bvec4` &value)
Specify value for `bvec4` uniform. [More...](#)

void `setUniform` (const std::string &name, const `Gsl::Mat3` &matrix)
Specify value for `mat3` matrix. [More...](#)

void `setUniform` (const std::string &name, const `Gsl::Mat4` &matrix)
Specify value for `mat4` matrix. [More...](#)

void `setUniform` (const std::string &name, const `Texture` &texture)
Specify a texture as `sampler2D` uniform. [More...](#)

void `setUniform` (const std::string &name, `CurrentTextureType`)
Specify current texture as `sampler2D` uniform. [More...](#)

void `setUniformArray` (const std::string &name, const float *scalar)
Specify values for `float[]` array uniform. [More...](#)

void `setUniformArray` (const std::string &name, const `Gsl::Vec2` &vector)
Specify values for `vec2[]` array uniform. [More...](#)

void `setUniformArray` (const std::string &name, const `Gsl::Vec3` &vector)
Specify values for `vec3[]` array uniform. [More...](#)

void `setUniformArray` (const std::string &name, const `Gsl::Vec4` &vector)
Specify values for `vec4[]` array uniform. [More...](#)

void `setUniformArray` (const std::string &name, const `Gsl::Mat3` &matrix)
Specify values for `mat3[]` array uniform. [More...](#)

void **setUniformArray** (const std::string &name, const **Gsl::Mat4**
Specify values for `mat4[]` array uniform. [More...](#)

void **setParameter** (const std::string &name, float x)
Change a float parameter of the shader. [More...](#)

void **setParameter** (const std::string &name, float x, float y)
Change a 2-components vector parameter of the shader. [M](#)

void **setParameter** (const std::string &name, float x, float y, float z)
Change a 3-components vector parameter of the shader. [M](#)

void **setParameter** (const std::string &name, float x, float y, float z, float w)
Change a 4-components vector parameter of the shader. [M](#)

void **setParameter** (const std::string &name, const **Vector2f** &vector)
Change a 2-components vector parameter of the shader. [M](#)

void **setParameter** (const std::string &name, const **Vector3f** &vector)
Change a 3-components vector parameter of the shader. [M](#)

void **setParameter** (const std::string &name, const **Color** &color)
Change a color parameter of the shader. [More...](#)

void **setParameter** (const std::string &name, const **Transform** &transform)
Change a matrix parameter of the shader. [More...](#)

void **setParameter** (const std::string &name, const **Texture** &texture)
Change a texture parameter of the shader. [More...](#)

void **setParameter** (const std::string &name, **CurrentTextureType** type)
Change a texture parameter of the shader. [More...](#)

unsigned int **getNativeHandle** () const
Get the underlying OpenGL handle of the shader. [More...](#)

Static Public Member Functions

static void **bind** (const **Shader** *shader)

Bind a shader for rendering. More...

static bool **isAvailable** ()

Tell whether or not the system supports shaders. More...

static bool **isGeometryAvailable** ()

Tell whether or not the system supports geometry shaders. M

Static Public Attributes

static CurrentTextureType CurrentTexture

Represents the texture of the object being drawn.

Static Private Member Functions

```
static void ensureGLContext ()
```

Empty function for ABI compatibility, use acquireTransientContext()

Detailed Description

Shader class (vertex, geometry and fragment)

Shaders are programs written using a specific language, executed directly allowing to apply real-time operations to the rendered entities.

There are three kinds of shaders:

- Vertex shaders, that process vertices
- Geometry shaders, that process primitives
- Fragment (pixel) shaders, that process pixels

A `sf::Shader` can be composed of either a vertex shader alone, a geometry shader alone, or any combination of them. (see the variants of the `load` function)

Shaders are written in GLSL, which is a C-like language dedicated to rendering. You will need to learn its basics before writing your own shaders for SFML.

Like any C/C++ program, a GLSL shader has its own variables called *uniforms*. In a C++ application, `sf::Shader` handles different types of uniforms:

- scalars: `float`, `int`, `bool`
- vectors (2, 3 or 4 components)
- matrices (3x3 or 4x4)
- samplers (textures)

Some SFML-specific types can be converted:

- `sf::Color` as a 4D vector (`vec4`)

- `sf::Transform` as matrices (`mat3` or `mat4`)

Every uniform variable in a shader can be set through one of the `s` overloads. For example, if you have a shader with the following uniforms:

```
uniform float offset;
uniform vec3 point;
uniform vec4 color;
uniform mat4 matrix;
uniform sampler2D overlay;
uniform sampler2D current;
```

You can set their values from C++ code as follows, using the types defined in `sf::Shader`:

```
shader.setUniform("offset", 2.f);
shader.setUniform("point", sf::Vector3f(0.5f, 0.8f, 0.3f));
shader.setUniform("color", sf::Gsl::Vec4(color));           // color is a sf::Color
shader.setUniform("matrix", sf::Gsl::Mat4(transform));      // transform is a sf::Transform
shader.setUniform("overlay", texture);                      // texture is a sf::Texture
shader.setUniform("current", sf::Shader::CurrentTexture);
```

The old `setParameter()` overloads are deprecated and will be removed in favor of their `setUniform()` equivalents instead.

The special `Shader::CurrentTexture` argument maps the given `sampler2D` to the object being drawn (which cannot be known in advance).

To apply a shader to a drawable, you must pass it as an additional parameter to the `draw()` function:

```
window.draw(sprite, &shader);
```

... which is in fact just a shortcut for this:

```
sf::RenderStates states;
states.shader = &shader;
window.draw(sprite, states);
```

In the code above we pass a pointer to the shader, because it may be null

Shaders can be used on any drawable, but some combinations are not vertex shader on a `sf::Sprite` is limited because there are only 4 vertices subdivided in order to apply wave effects. Another bad example is a font texture: the texture of the text is not the actual text that you see on screen, it is the characters of the font in an arbitrary order; thus, texture lookups on pixels will not give you the expected result.

Shaders can also be used to apply global post-effects to the current context (see the `sf::PostFx` class in SFML 1). This can be done in two different ways:

- draw everything to a `sf::RenderTarget`, then draw it to the main target
- draw everything directly to the main target, then use `sf::Texture::update` to a texture and draw it to the main target using the shader

The first technique is more optimized because it doesn't involve retrieving memory, but the second one doesn't impact the rendering process and without impacting all the code.

Like `sf::Texture` that can be used as a raw OpenGL texture, `sf::Shader` can be used as a raw OpenGL shader for custom OpenGL geometry.

```
sf::Shader::bind(&shader);
... render OpenGL geometry ...
sf::Shader::bind(NULL);
```

See also

`sf::Glsl`

Definition at line 52 of file `Shader.hpp`.

Member Enumeration Documentation

enum sf::Shader::Type

Types of shaders.

Enumerator

Vertex	Vertex shader
Geometry	Geometry shader.
Fragment	Fragment (pixel) shader.

Definition at line 60 of file `Shader.hpp`.

Constructor & Destructor Documentation

sf::Shader::Shader()

Default constructor.

This constructor creates an invalid shader.

sf::Shader::~Shader()

Destructor.

Member Function Documentation

static void sf::Shader::bind (const **Shader * shader)**

Bind a shader for rendering.

This function is not part of the graphics API, it mustn't be used when drawing with OpenGL. It can only be used if you mix sf::Shader with OpenGL code.

```
sf::Shader s1, s2;  
...  
sf::Shader::bind(&s1);  
// draw OpenGL stuff that use s1...  
sf::Shader::bind(&s2);  
// draw OpenGL stuff that use s2...  
sf::Shader::bind(NULL);  
// draw OpenGL stuff that use no shader...
```

Parameters

shader Shader to bind, can be null to use no shader

unsigned int sf::Shader::getNativeHandle () const

Get the underlying OpenGL handle of the shader.

You shouldn't need to use this function, unless you have very specific needs that OpenGL doesn't support, or implement a temporary workaround until a bug is fixed.

Returns

OpenGL handle of the shader or 0 if not yet loaded

static bool sf::Shader::isAvailable ()

Tell whether or not the system supports shaders.

This function should always be called before using the shader feature; any attempt to use `sf::Shader` will fail.

Returns

True if shaders are supported, false otherwise

static bool sf::Shader::isGeometryAvailable ()

Tell whether or not the system supports geometry shaders.

This function should always be called before using the geometry shader feature; any attempt to use `sf::Shader` geometry shader features will fail.

This function can only return true if `isAvailable()` would also return true, as geometry shaders must be supported in order for geometry shaders to be supported as well.

Note: The first call to this function, whether by your code or SFML will result in a warning message.

Returns

True if geometry shaders are supported, false otherwise

bool sf::Shader::loadFromFile (const std::string & filename, Type type)

Load the vertex, geometry or fragment shader from a file.

This function loads a single shader, vertex, geometry or fragment, ideally. The source must be a text file containing a valid shader in GLSL language. GLSL is a C-like language dedicated to OpenGL shaders; you'll probably need to read a good documentation for it before writing your own shaders.

Parameters

filename Path of the vertex, geometry or fragment shader file to load
type Type of shader (vertex, geometry or fragment)

Returns

True if loading succeeded, false if it failed

See also

[loadFromMemory](#), [loadFromStream](#)

```
bool sf::Shader::loadFromFile ( const std::string & vertexShaderFilename,
                               const std::string & fragmentShaderFilename )
```

Load both the vertex and fragment shaders from files.

This function loads both the vertex and the fragment shaders. If one of them is left empty (the valid shader is unloaded). The sources must be text files in GLSL language. GLSL is a C-like language dedicated to OpenGL shaders; you'll probably need to read a good documentation for it before writing your own shaders.

Parameters

vertexShaderFilename Path of the vertex shader file to load
fragmentShaderFilename Path of the fragment shader file to load

Returns

True if loading succeeded, false if it failed

See also

[loadFromMemory](#), [loadFromStream](#)

```
bool sf::Shader::loadFromFile ( const std::string & vertexShaderFile,
                                const std::string & geometryShaderFile,
                                const std::string & fragmentShaderFile )

```

Load the vertex, geometry and fragment shaders from files.

This function loads the vertex, geometry and fragment shaders. If one of them is left empty (the valid shader is unloaded). The sources must be text files in GLSL language. GLSL is a C-like language dedicated to OpenGL shaders; you can find documentation for it before writing your own shaders.

Parameters

vertexShaderFilename Path of the vertex shader file to load
geometryShaderFilename Path of the geometry shader file to load
fragmentShaderFilename Path of the fragment shader file to load

Returns

True if loading succeeded, false if it failed

See also

[loadFromMemory](#), [loadFromStream](#)

```
bool sf::Shader::loadFromMemory ( const std::string & shader,
                                  Type type )

```

)

Load the vertex, geometry or fragment shader from a source code in memory.

This function loads a single shader, vertex, geometry or fragment, ideally. The source code must be a valid shader in GLSL language. GLSL is a C-like language dedicated to OpenGL shaders; you'll probably need to read a good documentation about it before writing your own shaders.

Parameters

- shader** String containing the source code of the shader
- type** Type of shader (vertex, geometry or fragment)

Returns

True if loading succeeded, false if it failed

See also

[loadFromFile](#), [loadFromStream](#)

```
bool sf::Shader::loadFromMemory ( const std::string & vertexShaderSource,
                                  const std::string & fragmentShaderSource )

```

Load both the vertex and fragment shaders from source codes in memory.

This function loads both the vertex and the fragment shaders. If one of them is left empty (the valid shader is unloaded). The sources must be valid shader source codes in memory. GLSL is a C-like language dedicated to OpenGL shaders; you'll probably need to read a good documentation about it before writing your own shaders.

Parameters

- vertexShader** String containing the source code of the vertex shader
- fragmentShader** String containing the source code of the fragment shader

fragmentShader String containing the source code of the fragment

Returns

True if loading succeeded, false if it failed

See also

[loadFromFile](#), [loadFromStream](#)

```
bool sf::Shader::loadFromMemory ( const std::string & vertexShader,
                                  const std::string & geometryShader,
                                  const std::string & fragmentShader
                                )
```

Load the vertex, geometry and fragment shaders from source codes in memory.

This function loads the vertex, geometry and fragment shaders. If one of them is left empty (the valid shader is unloaded). The sources must be valid shader code in a C-like language dedicated to OpenGL shaders; you'll probably need to learn it before writing your own shaders.

Parameters

vertexShader String containing the source code of the vertex shader

geometryShader String containing the source code of the geometry shader

fragmentShader String containing the source code of the fragment shader

Returns

True if loading succeeded, false if it failed

See also

[loadFromFile](#), [loadFromStream](#)

```
bool sf::Shader::loadFromStream ( InputStream & stream,  
                                Type          type  
                                )
```

Load the vertex, geometry or fragment shader from a custom stream.

This function loads a single shader, vertex, geometry or fragment, ideally. The source code must be a valid shader in GLSL language. GLSL is a C-like language dedicated to OpenGL shaders; you'll probably need to read a good documentation for it before writing your own shaders.

Parameters

stream Source stream to read from
type Type of shader (vertex, geometry or fragment)

Returns

True if loading succeeded, false if it failed

See also

[loadFromFile](#), [loadFromMemory](#)

```
bool sf::Shader::loadFromStream ( InputStream & vertexShaderStream,  
                                InputStream & fragmentShaderStream  
                                )
```

Load both the vertex and fragment shaders from custom streams.

This function loads both the vertex and the fragment shaders. If one of them is left empty (the valid shader is unloaded). The source codes must be valid GLSL. GLSL is a C-like language dedicated to OpenGL shaders; you'll probably need to read a good documentation for it before writing your own shaders.

Parameters

vertexShaderStream Source stream to read the vertex shader file
fragmentShaderStream Source stream to read the fragment shader file

Returns

True if loading succeeded, false if it failed

See also

[loadFromFile](#), [loadFromMemory](#)

```
bool sf::Shader::loadFromStream ( InputStream & vertexShaderStream,
                                 InputStream & geometryShaderStream,
                                 InputStream & fragmentShaderStream )
```

Load the vertex, geometry and fragment shaders from custom streams.

This function loads the vertex, geometry and fragment shaders. If one of them is left empty (the valid shader is unloaded). The source codes must be valid GLSL code. GLSL is a C-like language dedicated to OpenGL shaders; you'll find more documentation for it before writing your own shaders.

Parameters

vertexShaderStream Source stream to read the vertex shader file
geometryShaderStream Source stream to read the geometry shader file
fragmentShaderStream Source stream to read the fragment shader file

Returns

True if loading succeeded, false if it failed

See also

[loadFromFile](#), [loadFromMemory](#)

```
void sf::Shader::setParameter ( const std::string & name,
                            float x
                            )
```

Change a float parameter of the shader.

Deprecated:

Use `setUniform(const std::string&, float)` instead.

```
void sf::Shader::setParameter ( const std::string & name,
                            float x,
                            float y
                            )
```

Change a 2-components vector parameter of the shader.

Deprecated:

Use `setUniform(const std::string&, const Gsl::Vec2&)` instead.

```
void sf::Shader::setParameter ( const std::string & name,
                            float x,
                            float y,
                            float z
                            )
```

Change a 3-components vector parameter of the shader.

Deprecated:

Use `setUniform(const std::string&, const Gsl::Vec3&)` instead.

```
void sf::Shader::setParameter ( const std::string & name,
                            float           x,
                            float           y,
                            float           z,
                            float           w
                        )
```

Change a 4-components vector parameter of the shader.

Deprecated:

Use `setUniform(const std::string&, const Gsl::Vec4&)` instead.

```
void sf::Shader::setParameter ( const std::string & name,
                            const Vector2f & vector
                        )
```

Change a 2-components vector parameter of the shader.

Deprecated:

Use `setUniform(const std::string&, const Gsl::Vec2&)` instead.

```
void sf::Shader::setParameter ( const std::string & name,
                            const Vector3f & vector
                        )
```

Change a 3-components vector parameter of the shader.

Deprecated:

| Use `setUniform(const std::string&, const Gsl::Vec3&)` instead.

```
void sf::Shader::setParameter (const std::string & name,  
                             const Color & color  
)
```

Change a color parameter of the shader.

Deprecated:

| Use `setUniform(const std::string&, const Gsl::Vec4&)` instead.

```
void sf::Shader::setParameter (const std::string & name,  
                             const Transform & transform  
)
```

Change a matrix parameter of the shader.

Deprecated:

| Use `setUniform(const std::string&, const Gsl::Mat4&)` instead.

```
void sf::Shader::setParameter (const std::string & name,  
                             const Texture & texture  
)
```

Change a texture parameter of the shader.

Deprecated:

| Use `setUniform(const std::string&, const Texture&)` instead.

```
void sf::Shader::setParameter ( const std::string & name,
                           CurrentTextureType
)
```

Change a texture parameter of the shader.

Deprecated:

Use `setUniform(const std::string&, CurrentTextureType)` instead.

```
void sf::Shader::setUniform ( const std::string & name,
                           float x
)
```

Specify value for `float` uniform.

Parameters

name Name of the uniform variable in GLSL

x Value of the float scalar

```
void sf::Shader::setUniform ( const std::string & name,
                           const Glsl::Vec2 & vector
)
```

Specify value for `vec2` uniform.

Parameters

name Name of the uniform variable in GLSL

vector Value of the vec2 vector

```
void sf::Shader::setUniform ( const std::string & name,
                           const Glsl::Vec3 & vector
)
```

Specify value for `vec3` uniform.

Parameters

name Name of the uniform variable in GLSL
vector Value of the `vec3` vector

```
void sf::Shader::setUniform ( const std::string & name,
                           const Glsl::Vec4 & vector
)
```

Specify value for `vec4` uniform.

This overload can also be called with `sf::Color` objects that are converted.

It is important to note that the components of the color are normalized by the shader. Therefore, they are converted from range [0 .. 255] to range [0.0 .. 1.0]. `sf::Color(255, 127, 0, 255)` will be transformed to a `vec4(1.0, 0.5, 0.0, 1.0)`.

Parameters

name Name of the uniform variable in GLSL
vector Value of the `vec4` vector

```
void sf::Shader::setUniform ( const std::string & name,
                           int x
)
```

Specify value for `int` uniform.

Parameters

name Name of the uniform variable in GLSL
x Value of the int scalar

```
void sf::Shader::setUniform ( const std::string & name,  
                            const Glsl::ivec2 & vector  
                          )
```

Specify value for `ivec2` uniform.

Parameters

name Name of the uniform variable in GLSL
vector Value of the ivec2 vector

```
void sf::Shader::setUniform ( const std::string & name,  
                            const Glsl::ivec3 & vector  
                          )
```

Specify value for `ivec3` uniform.

Parameters

name Name of the uniform variable in GLSL
vector Value of the ivec3 vector

```
void sf::Shader::setUniform ( const std::string & name,  
                            const Glsl::ivec4 & vector  
                          )
```

```
)
```

Specify value for `ivec4` uniform.

This overload can also be called with `sf::Color` objects that are converted.

If color conversions are used, the `ivec4` uniform in GLSL will hold the value of the `sf::Color` instance. For example, `sf::Color(255, 127, 0, 255)` is mapped to `ivec4(255, 127, 0, 255)`.

Parameters

name Name of the uniform variable in GLSL

vector Value of the `ivec4` vector

```
void sf::Shader::setUniform ( const std::string & name,  
                           bool x  
)
```

Specify value for `bool` uniform.

Parameters

name Name of the uniform variable in GLSL

x Value of the `bool` scalar

```
void sf::Shader::setUniform ( const std::string & name,  
                           const Gsl::Bvec2 & vector  
)
```

Specify value for `bvec2` uniform.

Parameters

name Name of the uniform variable in GLSL
vector Value of the bvec2 vector

```
void sf::Shader::setUniform ( const std::string & name,
                            const Gsl::Bvec3 & vector
                           )
```

Specify value for `bvec3` uniform.

Parameters

name Name of the uniform variable in GLSL
vector Value of the bvec3 vector

```
void sf::Shader::setUniform ( const std::string & name,
                            const Gsl::Bvec4 & vector
                           )
```

Specify value for `bvec4` uniform.

Parameters

name Name of the uniform variable in GLSL
vector Value of the bvec4 vector

```
void sf::Shader::setUniform ( const std::string & name,
                            const Gsl::Mat3 & matrix
                           )
```

Specify value for `mat3` matrix.

Parameters

name Name of the uniform variable in GLSL

matrix Value of the mat3 matrix

```
void sf::Shader::setUniform ( const std::string & name,  
                            const Glsl::Mat4 & matrix  
                          )
```

Specify value for `mat4` matrix.

Parameters

name Name of the uniform variable in GLSL

matrix Value of the mat4 matrix

```
void sf::Shader::setUniform ( const std::string & name,  
                            const Texture & texture  
                          )
```

Specify a texture as `sampler2D` uniform.

name is the name of the variable to change in the shader. The corresponding variable must be a 2D texture (`sampler2D` GLSL type).

Example:

```
uniform sampler2D the_texture; // this is the variable in the shader  
  
sf::Texture texture;  
...  
shader.setUniform("the_texture", texture);
```

It is important to note that *texture* must remain alive as long as the shader.

internally.

To use the texture of the object being drawn, which cannot be known at draw time, use the special value `sf::Shader::CurrentTexture`:

```
shader.setUniform("the_texture", sf::Shader::CurrentTexture);
```

Parameters

name Name of the texture in the shader
texture Texture to assign

```
void sf::Shader::setUniform ( const std::string & name,  
                           CurrentTextureType  
                           )
```

Specify current texture as sampler2D uniform.

This overload maps a shader texture variable to the texture of the object known in advance. The second argument must be `sf::Shader::Current` parameter in the shader must be a 2D texture (`sampler2D` GLSL type).

Example:

```
uniform sampler2D current; // this is the variable in the shader  
shader.setUniform("current", sf::Shader::CurrentTexture);
```

Parameters

name Name of the texture in the shader

```
    std::size_t length  
)
```

Specify values for `float[]` array uniform.

Parameters

name Name of the uniform variable in GLSL
scalarArray pointer to array of `float` values
length Number of elements in the array

```
void sf::Shader::setUniformArray ( const std::string & name,  
                                  const Glsl::Vec2 * vectorArray,  
                                  std::size_t length  
)
```

Specify values for `vec2[]` array uniform.

Parameters

name Name of the uniform variable in GLSL
vectorArray pointer to array of `vec2` values
length Number of elements in the array

```
void sf::Shader::setUniformArray ( const std::string & name,  
                                  const Glsl::Vec3 * vectorArray,  
                                  std::size_t length  
)
```

Specify values for `vec3[]` array uniform.

Parameters

name Name of the uniform variable in GLSL
vectorArray pointer to array of `vec3` values
length Number of elements in the array

```
void sf::Shader::setUniformArray ( const std::string & name,  
                                  const Gsl::Vec4 * vectorArray,  
                                  std::size_t length  
    )
```

Specify values for `vec4[]` array uniform.

Parameters

name Name of the uniform variable in GLSL
vectorArray pointer to array of `vec4` values
length Number of elements in the array

```
void sf::Shader::setUniformArray ( const std::string & name,  
                                  const Gsl::Mat3 * matrixArray,  
                                  std::size_t length  
    )
```

Specify values for `mat3[]` array uniform.

Parameters

name Name of the uniform variable in GLSL
matrixArray pointer to array of `mat3` values
length Number of elements in the array

```
void sf::Shader::setUniformArray ( const std::string & name,
```

```
    const Gsl::Mat4 * matrixArray,  
    std::size_t length  
)
```

Specify values for `mat4[]` array uniform.

Parameters

- name** Name of the uniform variable in GLSL
- matrixArray** pointer to array of `mat4` values
- length** Number of elements in the array

Member Data Documentation

CurrentTextureType sf::Shader::CurrentTexture

Represents the texture of the object being drawn.

See also

[setUniform\(const std::string&, CurrentTextureType\)](#)

Definition at line [82](#) of file [Shader.hpp](#).

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

- [Shader.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Shader::CurrentTextureType Struct Reference

Special type that can be passed to `setUniform()`, and that represents the

[More...](#)

```
#include <Shader.hpp>
```

Detailed Description

Special type that can be passed to `setUniform()`, and that represents the t

See also

`setUniform(const std::string&, CurrentTextureType)`

Definition at line 74 of file `Shader.hpp`.

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

- `Shader.hpp`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [Protected Member Functions](#) | [List of all members](#)

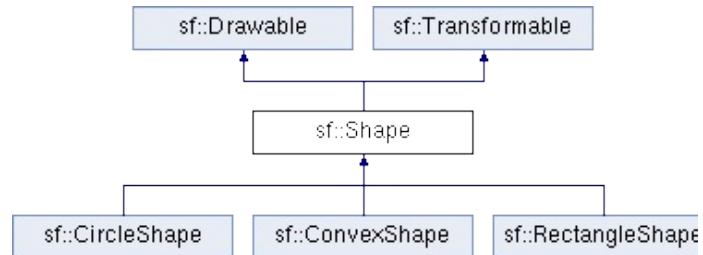
sf::Shape Class Reference abstract

[Graphics module](#)

Base class for textured shapes with outline. [More...](#)

```
#include <Shape.hpp>
```

Inheritance diagram for sf::Shape:



Public Member Functions

virtual ~Shape ()
Virtual destructor. [More...](#)

void setTexture (const Texture *texture, bool resetRect=false)
Change the source texture of the shape. [More...](#)

void setTextureRect (const IntRect &rect)
Set the sub-rectangle of the texture that the shape will use.

void setFillColor (const Color &color)
Set the fill color of the shape. [More...](#)

void setOutlineColor (const Color &color)
Set the outline color of the shape. [More...](#)

void setOutlineThickness (float thickness)
Set the thickness of the shape's outline. [More...](#)

const Texture * getTexture () const
Get the source texture of the shape. [More...](#)

const IntRect & getTextureRect () const
Get the sub-rectangle of the texture displayed by the shape.

const Color & getFillColor () const
Get the fill color of the shape. [More...](#)

const Color & getOutlineColor () const
Get the outline color of the shape. [More...](#)

float getOutlineThickness () const

Get the outline thickness of the shape. [More...](#)

virtual std::size_t **getPointCount () const** =0
Get the total number of points of the shape. [More...](#)

virtual Vector2f **getPoint (std::size_t index) const** =0
Get a point of the shape. [More...](#)

FloatRect **getLocalBounds () const**
Get the local bounding rectangle of the entity. [More..](#)

FloatRect **getGlobalBounds () const**
Get the global (non-minimal) bounding rectangle of the entity. [More..](#)

void **setPosition (float x, float y)**
set the position of the object [More...](#)

void **setPosition (const Vector2f &position)**
set the position of the object [More...](#)

void **setRotation (float angle)**
set the orientation of the object [More...](#)

void **setScale (float factorX, float factorY)**
set the scale factors of the object [More...](#)

void **setScale (const Vector2f &factors)**
set the scale factors of the object [More...](#)

void **setOrigin (float x, float y)**
set the local origin of the object [More...](#)

void **setOrigin (const Vector2f &origin)**
set the local origin of the object [More...](#)

`const Vector2f & getPosition () const`
get the position of the object [More...](#)

`float getRotation () const`
get the orientation of the object [More...](#)

`const Vector2f & getScale () const`
get the current scale of the object [More...](#)

`const Vector2f & getOrigin () const`
get the local origin of the object [More...](#)

`void move (float offsetX, float offsetY)`
Move the object by a given offset. [More...](#)

`void move (const Vector2f &offset)`
Move the object by a given offset. [More...](#)

`void rotate (float angle)`
Rotate the object. [More...](#)

`void scale (float factorX, float factorY)`
Scale the object. [More...](#)

`void scale (const Vector2f &factor)`
Scale the object. [More...](#)

`const Transform & getTransform () const`
get the combined transform of the object [More...](#)

`const Transform & getInverseTransform () const`
get the inverse of the combined transform of the obje

Protected Member Functions

`Shape ()`

Default constructor. [More...](#)

`void update ()`

Recompute the internal geometry of the shape. [More...](#)

Detailed Description

Base class for textured shapes with outline.

`sf::Shape` is a drawable class that allows to define and display a custom c

It's only an abstract base, it needs to be specialized for concrete types of polygon, star, ...).

In addition to the attributes provided by the specialized shape classes, it has the following attributes:

- a texture
- a texture rectangle
- a fill color
- an outline color
- an outline thickness

Each feature is optional, and can be disabled easily:

- the texture can be null
- the fill/outline colors can be `sf::Color::Transparent`
- the outline thickness can be zero

You can write your own derived shape class, there are only two virtual functions to implement:

- `getPointCount` must return the number of points of the shape
- `getPoint` must return the points of the shape

See also

`sf::RectangleShape`, `sf::CircleShape`, `sf::ConvexShape`, `sf::Transform`

Definition at line 44 of file `Shape.hpp`.

Constructor & Destructor Documentation

virtual sf::Shape::~Shape()

Virtual destructor.

sf::Shape::Shape()

Default constructor.

Member Function Documentation

`const Color& sf::Shape::getFillColor() const`

Get the fill color of the shape.

Returns

Fill color of the shape

See also

[setFillColor](#)

`FloatRect sf::Shape::getGlobalBounds() const`

Get the global (non-minimal) bounding rectangle of the entity.

The returned rectangle is in global coordinates, which means it returns the bounds of the shape in the global 2D world's coordinate system. This function does not necessarily return the *minimal* bounding rectangle covers all the vertices (but possibly more). This allows bounds as a first check; you may want to use more precise checks on top of this one.

Returns

Global bounding rectangle of the entity

const Transform& sf::Transformable::getInverseTransform() const

get the inverse of the combined transform of the object

Returns

Inverse of the combined transformations applied to the object

See also

[getTransform](#)

FloatRect sf::Shape::getLocalBounds() const

Get the local bounding rectangle of the entity.

The returned rectangle is in local coordinates, which means that (translation, rotation, scale, ...) that are applied to the entity. In other bounds of the entity in the entity's coordinate system.

Returns

Local bounding rectangle of the entity

const Vector2f& sf::Transformable::getOrigin() const

get the local origin of the object

Returns

Current origin

See also

[setOrigin](#)

const Color& sf::Shape::getOutlineColor() const

Get the outline color of the shape.

Returns

Outline color of the shape

See also

[setOutlineColor](#)

float sf::Shape::getOutlineThickness() const

Get the outline thickness of the shape.

Returns

Outline thickness of the shape

See also

[setOutlineThickness](#)

virtual Vector2f sf::Shape::getPoint(std::size_t index) const

Get a point of the shape.

The returned point is in local coordinates, that is, the shape's transforms taken into account. The result is undefined if *index* is out of the valid range [0 .. `getPointCount()` - 1].

Parameters

index Index of the point to get, in range [0 .. `getPointCount()` - 1]

Returns

index-th point of the shape

See also

[getPointCount](#)

Implemented in [sf::ConvexShape](#), [sf::CircleShape](#), and [sf::RectangleShape](#)

virtual std::size_t sf::Shape::getPointCount() const

Get the total number of points of the shape.

Returns

Number of points of the shape

See also

[getPoint](#)

Implemented in [sf::CircleShape](#), [sf::RectangleShape](#), and [sf::ConvexShape](#)

const Vector2f& sf::Transformable::getPosition() const

get the position of the object

Returns

Current position

See also

[setPosition](#)

`float sf::Transformable::getRotation() const`

get the orientation of the object

The rotation is always in the range [0, 360].

Returns

Current rotation, in degrees

See also

[setRotation](#)

`const Vector2f& sf::Transformable::getScale() const`

get the current scale of the object

Returns

Current scale factors

See also

[setScale](#)

`const Texture* sf::Shape::getTexture() const`

Get the source texture of the shape.

If the shape has no source texture, a NULL pointer is returned. This means that you can't modify the texture when you retrieve it with this function.

Returns

Pointer to the shape's texture

See also

[setTexture](#)

const IntRect& sf::Shape::getTextureRect() const

Get the sub-rectangle of the texture displayed by the shape.

Returns

Texture rectangle of the shape

See also

[setTextureRect](#)

const Transform& sf::Transformable::getTransform() const

get the combined transform of the object

Returns

Transform combining the position/rotation/scale/origin of the object

See also

[getInverseTransform](#)

void sf::Transformable::move(float offsetX, float offsetY)

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition` equivalent to the following code:

```
sf::Vector2f pos = object.getPosition();
object.setPosition(pos.x + offsetX, pos.y + offsetY);
```

Parameters

offsetX X offset

offsetY Y offset

See also

[setPosition](#)

void sf::Transformable::move (const Vector2f & offset)

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition` equivalent to the following code:

```
object.setPosition(object.getPosition() + offset);
```

Parameters

offset Offset

See also

[setPosition](#)

void sf::Transformable::rotate (float angle)

Rotate the object.

This function adds to the current rotation of the object, unlike setRotation equivalent to the following code:

```
object.setRotation(object.getRotation() + angle);
```

Parameters

angle Angle of rotation, in degrees

```
void sf::Transformable::scale ( float factorX,  
                             float factorY  
                           )
```

Scale the object.

This function multiplies the current scale of the object, unlike setScale equivalent to the following code:

```
sf::Vector2f scale = object.getScale();  
object.setScale(scale.x * factorX, scale.y * factorY);
```

Parameters

factorX Horizontal scale factor

factorY Vertical scale factor

See also

[setScale](#)

```
void sf::Transformable::scale ( const Vector2f & factor )
```

Scale the object.

This function multiplies the current scale of the object, unlike `setScale` equivalent to the following code:

```
sf::Vector2f scale = object.getScale();
object.setScale(scale.x * factor.x, scale.y * factor.y);
```

Parameters

factor Scale factors

See also

`setScale`

void sf::Shape::setFillColor (const Color & color)

Set the fill color of the shape.

This color is modulated (multiplied) with the shape's texture if any. It can or change its global opacity. You can use `sf::Color::Transparent` to transparent, and have the outline alone. By default, the shape's fill color

Parameters

color New color of the shape

See also

`getFillColor`, `setOutlineColor`

void sf::Transformable::setOrigin (float x, float y)

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transform

Parameters

- x** X coordinate of the new origin
- y** Y coordinate of the new origin

See also

[getOrigin](#)

void sf::Transformable::setOrigin (const Vector2f & origin)

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transform

Parameters

- origin** New origin

See also

[getOrigin](#)

void sf::Shape::setOutlineColor (const Color & color)

Set the outline color of the shape.

By default, the shape's outline color is opaque white.

Parameters

color New outline color of the shape

See also

[getOutlineColor](#), [setFillColor](#)

void sf::Shape::setOutlineThickness (float thickness)

Set the thickness of the shape's outline.

Note that negative values are allowed (so that the outline expands toward the center). Using zero disables the outline. By default, the outline thickness is 0.

Parameters

thickness New outline thickness

See also

[getOutlineThickness](#)

void sf::Transformable::setPosition (float x, float y)

set the position of the object

This function completely overwrites the previous position. See the [move](#) function instead. The default position of a transformable

Parameters

x X coordinate of the new position

y Y coordinate of the new position

See also

[move](#), [getPosition](#)

void sf::Transformable::setPosition (const [Vector2f](#) & position)

set the position of the object

This function completely overwrites the previous position. See the [move](#) on the previous position instead. The default position of a transformable

Parameters

position New position

See also

[move](#), [getPosition](#)

void sf::Transformable::setRotation (float angle)

set the orientation of the object

This function completely overwrites the previous rotation. See the [rotate](#) on the previous rotation instead. The default rotation of a transformable

Parameters

angle New rotation, in degrees

See also

[rotate](#), [getRotation](#)

```
void sf::Transformable::setScale ( float factorX,  
                                 float factorY  
                               )
```

set the scale factors of the object

This function completely overwrites the previous scale. See the `scale` function for more information about scaling objects.

Parameters

factorX New horizontal scale factor

factorY New vertical scale factor

See also

scale, getScale

```
void sf::Transformable::setScale ( const Vector2f & factors )
```

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function for the previous scale instead. The default scale of a transformable object is

Parameters

factors New scale factors

See also

scale, getScale

```
void sf::Shape::setTexture ( const Texture * texture,  
                           bool                  resetRect = false )
```

)

Change the source texture of the shape.

The *texture* argument refers to a texture that must exist as long as the shape does. The shape doesn't store its own copy of the texture, but rather keeps a pointer to it. If the source texture is destroyed and the shape tries to use it, the texture can be NULL to disable texturing. If *resetRect* is true, the TextureRect of the shape will be automatically adjusted to the size of the new texture. If it is false, the texture rect will remain the same.

Parameters

texture New texture

resetRect Should the texture rect be reset to the size of the new texture?

See also

[getTexture](#), [setTextureRect](#)

void sf::Shape::setTextureRect(const IntRect & rect)

Set the sub-rectangle of the texture that the shape will display.

The texture rect is useful when you don't want to display the whole texture. By default, the texture rect covers the entire texture.

Parameters

rect Rectangle defining the region of the texture to display

See also

[getTextureRect](#), [setTexture](#)

void sf::Shape::update()

Recompute the internal geometry of the shape.

This function must be called by the derived class everytime the shape's either getPointCount or getPoint is different).

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

- [Shape.hpp](#)
-

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [List of all members](#)

sf::Sprite Class Reference

[Graphics module](#)

Drawable representation of a texture, with its own transformations, color, etc.

```
#include <Sprite.hpp>
```

Inheritance diagram for sf::Sprite:



Public Member Functions

`Sprite ()`

Default constructor. [More...](#)

`Sprite (const Texture &texture)`

Construct the sprite from a source texture. [More...](#)

`Sprite (const Texture &texture, const IntRect &rectangle)`

Construct the sprite from a sub-rectangle of a source

`void setTexture (const Texture &texture, bool resetRect=false)`

Change the source texture of the sprite. [More...](#)

`void setTextureRect (const IntRect &rectangle)`

Set the sub-rectangle of the texture that the sprite will

`void setColor (const Color &color)`

Set the global color of the sprite. [More...](#)

`const Texture * getTexture () const`

Get the source texture of the sprite. [More...](#)

`const IntRect & getTextureRect () const`

Get the sub-rectangle of the texture displayed by the

`const Color & getColor () const`

Get the global color of the sprite. [More...](#)

`FloatRect getLocalBounds () const`

Get the local bounding rectangle of the entity. [More...](#)

`FloatRect getGlobalBounds () const`

Get the global bounding rectangle of the entity. More

void **setPosition** (float x, float y)

set the position of the object More...

void **setPosition** (const **Vector2f** &position)

set the position of the object More...

void **setRotation** (float angle)

set the orientation of the object More...

void **setScale** (float factorX, float factorY)

set the scale factors of the object More...

void **setScale** (const **Vector2f** &factors)

set the scale factors of the object More...

void **setOrigin** (float x, float y)

set the local origin of the object More...

void **setOrigin** (const **Vector2f** &origin)

set the local origin of the object More...

const **Vector2f** & **getPosition** () const

get the position of the object More...

float **getRotation** () const

get the orientation of the object More...

const **Vector2f** & **getScale** () const

get the current scale of the object More...

const **Vector2f** & **getOrigin** () const

get the local origin of the object More...

```
void move (float offsetX, float offsetY)  
    Move the object by a given offset. More...
```

```
void move (const Vector2f &offset)  
    Move the object by a given offset. More...
```

```
void rotate (float angle)  
    Rotate the object. More...
```

```
void scale (float factorX, float factorY)  
    Scale the object. More...
```

```
void scale (const Vector2f &factor)  
    Scale the object. More...
```

```
const Transform & getTransform () const  
    get the combined transform of the object More...
```

```
const Transform & getInverseTransform () const  
    get the inverse of the combined transform of the obje
```

Detailed Description

Drawable representation of a texture, with its own transformations, color, etc.

`sf::Sprite` is a drawable class that allows to easily display a texture (or a part of it).

It inherits all the functions from `sf::Transformable`: position, rotation, scale properties such as the texture to use, the part of it to display, and some others such as the overall color of the sprite, or to get its bounding rectangle.

`sf::Sprite` works in combination with the `sf::Texture` class, which loads the given texture.

The separation of `sf::Sprite` and `sf::Texture` allows more flexibility and efficiency. `sf::Texture` is a heavy resource, and any operation on it is slow (often too slow to be used in a loop). On the other side, a `sf::Sprite` is a lightweight object which can use the pi-

nit with its own transformation/color/blending attributes.

It is important to note that the `sf::Sprite` instance doesn't copy the texture reference to it. Thus, a `sf::Texture` must not be destroyed while it is used by a function that uses a local `sf::Texture` instance for creating a sprite).

See also the note on coordinates and undistorted rendering in `sf::Transformable`.

Usage example:

```
// Declare and load a texture
sf::Texture texture;
texture.loadFromFile("texture.png");

// Create a sprite
sf::Sprite sprite;
sprite.setTexture(texture);
sprite.setTextureRect(sf::IntRect(10, 10, 50, 30));
sprite.setColor(sf::Color(255, 255, 255, 200));
```

```
sprite.setPosition(100, 25);  
// Draw it  
window.draw(sprite);
```

See also

[sf::Texture](#), [sf::Transformable](#)

Definition at line 47 of file [Sprite.hpp](#).

Constructor & Destructor Documentation

`sf::Sprite::Sprite()`

Default constructor.

Creates an empty sprite with no source texture.

`sf::Sprite::Sprite(const Texture & texture)`

Construct the sprite from a source texture.

Parameters

texture Source texture

See also

[setTexture](#)

`sf::Sprite::Sprite(const Texture & texture, const IntRect & rectangle)`

Construct the sprite from a sub-rectangle of a source texture.

Parameters

texture Source texture

rectangle Sub-rectangle of the texture to assign to the sprite

See also

[setTexture](#), [setTextureRect](#)

Member Function Documentation

const Color& sf::Sprite::getColor() const

Get the global color of the sprite.

Returns

Global color of the sprite

See also

[setColor](#)

FloatRect sf::Sprite::getGlobalBounds() const

Get the global bounding rectangle of the entity.

The returned rectangle is in global coordinates, which means it returns the bounds of the sprite in the global 2D world's coordinate system. This function applies all transformations (translation, rotation, scale, ...) that are applied to the entity.

Returns

Global bounding rectangle of the entity

const Transform& sf::Transformable::getInverseTransform() const

get the inverse of the combined transform of the object

Returns

Inverse of the combined transformations applied to the object

See also

[getTransform](#)

FloatRect sf::Sprite::getLocalBounds() const

Get the local bounding rectangle of the entity.

The returned rectangle is in local coordinates, which means that (translation, rotation, scale, ...) that are applied to the entity. In other bounds of the entity in the entity's coordinate system.

Returns

Local bounding rectangle of the entity

const Vector2f& sf::Transformable::getOrigin() const

get the local origin of the object

Returns

Current origin

See also

[setOrigin](#)

const Vector2f& sf::Transformable::getPosition() const

get the position of the object

Returns

Current position

See also

[setPosition](#)

float sf::Transformable::getRotation() const

get the orientation of the object

The rotation is always in the range [0, 360].

Returns

Current rotation, in degrees

See also

[setRotation](#)

const Vector2f& sf::Transformable::getScale() const

get the current scale of the object

Returns

Current scale factors

See also

[setScale](#)

const Texture* sf::Sprite::getTexture() const

Get the source texture of the sprite.

If the sprite has no source texture, a NULL pointer is returned. The **const** means that you can't modify the texture when you retrieve it with this function.

Returns

Pointer to the sprite's texture

See also

[setTexture](#)

const IntRect& sf::Sprite::getTextureRect() const

Get the sub-rectangle of the texture displayed by the sprite.

Returns

Texture rectangle of the sprite

See also

[setTextureRect](#)

const Transform& sf::Transformable::getTransform() const

get the combined transform of the object

Returns

Transform combining the position/rotation/scale/origin of the object

See also

[getInverseTransform](#)

```
void sf::Transformable::move ( float offsetX,  
                            float offsetY  
                          )
```

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition` which is equivalent to the following code:

```
sf::Vector2f pos = object.getPosition();  
object.setPosition(pos.x + offsetX, pos.y + offsetY);
```

Parameters

offsetX X offset

offsetY Y offset

See also

[setPosition](#)

```
void sf::Transformable::move ( const Vector2f & offset )
```

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition` which is equivalent to the following code:

```
object.setPosition(object.getPosition() + offset);
```

Parameters

offset Offset

See also

[setPosition](#)

void sf::Transformable::rotate (float angle)

Rotate the object.

This function adds to the current rotation of the object, unlike `setRotation`, equivalent to the following code:

```
object.setRotation(object.getRotation() + angle);
```

Parameters

angle Angle of rotation, in degrees

void sf::Transformable::scale (float factorX, float factorY)

Scale the object.

This function multiplies the current scale of the object, unlike `setScale`, equivalent to the following code:

```
sf::Vector2f scale = object.getScale();  
object.setScale(scale.x * factorX, scale.y * factorY);
```

Parameters

factorX Horizontal scale factor
factorY Vertical scale factor

See also

[setScale](#)

void sf::Transformable::scale (const Vector2f & factor)

Scale the object.

This function multiplies the current scale of the object, unlike `setScale` equivalent to the following code:

```
sf::Vector2f scale = object.getScale();
object.setScale(scale.x * factor.x, scale.y * factor.y);
```

Parameters

factor Scale factors

See also

[setScale](#)

void sf::Sprite::setColor (const Color & color)

Set the global color of the sprite.

This color is modulated (multiplied) with the sprite's texture. It can be change its global opacity. By default, the sprite's color is opaque white.

Parameters

color New color of the sprite

See also

[getColor](#)

```
void sf::Transformable::setOrigin ( float x,  
                                  float y  
 )
```

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transfor

Parameters

- x** X coordinate of the new origin
- y** Y coordinate of the new origin

See also

[getOrigin](#)

```
void sf::Transformable::setOrigin ( const Vector2f & origin )
```

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transfor

Parameters

- origin** New origin

See also

[getOrigin](#)

```
void sf::Transformable::setPosition ( float x,  
                                    float y  
                                )
```

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

- x** X coordinate of the new position
- y** Y coordinate of the new position

See also

[move](#), [getPosition](#)

```
void sf::Transformable::setPosition ( const Vector2f & position )
```

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

- position** New position

See also

[move](#), [getPosition](#)

void sf::Transformable::setRotation (float angle)

set the orientation of the object

This function completely overwrites the previous rotation. See the [rotate](#) on the previous rotation instead. The default rotation of a transformable object is 0 degrees.

Parameters

angle New rotation, in degrees

See also

[rotate](#), [getRotation](#)

void sf::Transformable::setScale (float factorX, float factorY)

set the scale factors of the object

This function completely overwrites the previous scale. See the [scale](#) on the previous scale instead. The default scale of a transformable object is (1, 1).

Parameters

factorX New horizontal scale factor

factorY New vertical scale factor

See also

[scale](#), [getScale](#)

```
void sf::Transformable::setScale ( const Vector2f & factors )
```

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function for more information about how it differs from this function.
The previous scale instead. The default scale of a transformable object is (1, 1).

Parameters

factors New scale factors

See also

[scale](#), [getScale](#)

```
void sf::Sprite::setTexture ( const Texture & texture,
                            bool resetRect = false
                          )
```

Change the source texture of the sprite.

The *texture* argument refers to a texture that must exist as long as the sprite does. The sprite doesn't store its own copy of the texture, but rather keeps a pointer to it. If the source texture is destroyed and the sprite tries to use it, the texture is automatically deallocated. If *resetRect* is true, the *TextureRect* property of the sprite is automatically reset to the size of the new texture. If it is false, the texture rect is left unchanged.

Parameters

texture New texture

resetRect Should the texture rect be reset to the size of the new texture?

See also

[getTexture](#), [setTextureRect](#)

void sf::Sprite::setTextureRect (const IntRect & rectangle)

Set the sub-rectangle of the texture that the sprite will display.

The texture rect is useful when you don't want to display the whole texture by default, the texture rect covers the entire texture.

Parameters

rectangle Rectangle defining the region of the texture to display

See also

[getTextureRect](#), [setTexture](#)

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

- [Sprite.hpp](#)

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [List of all members](#)

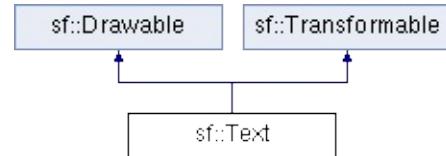
sf::Text Class Reference

Graphics module

Graphical text that can be drawn to a render target. [More...](#)

```
#include <Text.hpp>
```

Inheritance diagram for sf::Text:



Public Types

```
enum Style {  
    Regular = 0, Bold = 1 << 0, Italic = 1 << 1, Underlined = 1 << 2  
    StrikeThrough = 1 << 3  
}
```

Enumeration of the string drawing styles. [More...](#)

Public Member Functions

`Text ()`

Default constructor. [More...](#)

`Text (const String &string, const Font &font, unsigned size)`

Construct the text from a string, font and size. [More...](#)

`void setString (const String &string)`

Set the text's string. [More...](#)

`void setFont (const Font &font)`

Set the text's font. [More...](#)

`void setCharacterSize (unsigned int size)`

Set the character size. [More...](#)

`void setStyle (Uint32 style)`

Set the text's style. [More...](#)

`void setColor (const Color &color)`

Set the fill color of the text. [More...](#)

`void setFillColor (const Color &color)`

Set the fill color of the text. [More...](#)

`void setOutlineColor (const Color &color)`

Set the outline color of the text. [More...](#)

`void setOutlineThickness (float thickness)`

Set the thickness of the text's outline. [More...](#)

`const String & getString () const`

Get the text's string. [More...](#)

const **Font** * **getFont** () const
Get the text's font. [More...](#)

unsigned int **getCharacterSize** () const
Get the character size. [More...](#)

Uint32 **getStyle** () const
Get the text's style. [More...](#)

const **Color** & **getColor** () const
Get the fill color of the text. [More...](#)

const **Color** & **getFillColor** () const
Get the fill color of the text. [More...](#)

const **Color** & **getOutlineColor** () const
Get the outline color of the text. [More...](#)

float **getOutlineThickness** () const
Get the outline thickness of the text. [More...](#)

Vector2f **findCharacterPos** (std::size_t index) const
Return the position of the *index-th* character. [More...](#)

FloatRect **getLocalBounds** () const
Get the local bounding rectangle of the entity. [More..](#)

FloatRect **getGlobalBounds** () const
Get the global bounding rectangle of the entity. [More](#)

void **setPosition** (float x, float y)
set the position of the object [More...](#)

`void setPosition (const Vector2f &position)`
set the position of the object [More...](#)

`void setRotation (float angle)`
set the orientation of the object [More...](#)

`void setScale (float factorX, float factorY)`
set the scale factors of the object [More...](#)

`void setScale (const Vector2f &factors)`
set the scale factors of the object [More...](#)

`void setOrigin (float x, float y)`
set the local origin of the object [More...](#)

`void setOrigin (const Vector2f &origin)`
set the local origin of the object [More...](#)

`const Vector2f & getPosition () const`
get the position of the object [More...](#)

`float getRotation () const`
get the orientation of the object [More...](#)

`const Vector2f & getScale () const`
get the current scale of the object [More...](#)

`const Vector2f & getOrigin () const`
get the local origin of the object [More...](#)

`void move (float offsetX, float offsetY)`
Move the object by a given offset. [More...](#)

`void move (const Vector2f &offset)`
Move the object by a given offset. [More...](#)

`void rotate (float angle)`
Rotate the object. [More...](#)

`void scale (float factorX, float factorY)`
Scale the object. [More...](#)

`void scale (const Vector2f &factor)`
Scale the object. [More...](#)

`const Transform & getTransform () const`
get the combined transform of the object [More...](#)

`const Transform & getInverseTransform () const`
get the inverse of the combined transform of the obje

Detailed Description

Graphical text that can be drawn to a render target.

`sf::Text` is a drawable class that allows to easily display some text with a render target.

It inherits all the functions from `sf::Transformable`: position, rotation, scale properties such as the font to use, the character size, the font style (bold, italic), the global color and the text to display of course. It also provides convenience methods to calculate the graphical size of the text, or to get the global position of a given character.

`sf::Text` works in combination with the `sf::Font` class, which loads the glyphs (the characters) of a given font.

The separation of `sf::Font` and `sf::Text` allows more flexibility and better performance. As a `sf::Font` is a heavy resource, and any operation on it is slow (often too slow for real-time rendering), a `sf::Text` is a lightweight object which can combine the glyphs loaded by a `sf::Font` to display any text on a render target.

It is important to note that the `sf::Text` instance doesn't copy the font that it uses. Thus, a `sf::Font` must not be destructed while it is used by a `sf::Text` (as the `sf::Text` uses a local `sf::Font` instance for creating a text).

See also the note on coordinates and undistorted rendering in `sf::Transformable`.

Usage example:

```
// Declare and load a font
sf::Font font;
font.loadFromFile("arial.ttf");

// Create a text
```

```
sf::Text text("hello", font);
text.setCharacterSize(30);
text.setStyle(sf::Text::Bold);
text.setColor(sf::Color::Red);

// Draw it
window.draw(text);
```

See also

[sf::Font](#), [sf::Transformable](#)

Definition at line 48 of file [Text.hpp](#).

Member Enumeration Documentation

enum sf::Text::Style

Enumeration of the string drawing styles.

Enumerator	
Regular	Regular characters, no style.
Bold	Bold characters.
Italic	Italic characters.
Underlined	Underlined characters.
StrikeThrough	Strike through characters.

Definition at line 56 of file [Text.hpp](#).

Constructor & Destructor Documentation

sf::Text::Text()

Default constructor.

Creates an empty text.

```
sf::Text::Text( const String & string,
            const Font & font,
            unsigned int characterSize = 30
)
```

Construct the text from a string, font and size.

Note that if the used font is a bitmap font, it is not scalable, thus not all to use. This needs to be taken into consideration when setting the character size of a certain size, make sure the corresponding bitmap font that supports it.

Parameters

string	Text assigned to the string
font	Font used to draw the string
characterSize	Base size of characters, in pixels

Member Function Documentation

`Vector2f sf::Text::findCharacterPos (std::size_t index) const`

Return the position of the *index-th* character.

This function computes the visual position of a character from its index. The position is in global coordinates (translation, rotation, scale and origin). If the character's range, the position of the end of the string is returned.

Parameters

index Index of the character

Returns

Position of the character

`unsigned int sf::Text::getCharacterSize () const`

Get the character size.

Returns

Size of the characters, in pixels

See also

`setCharacterSize`

`const Color& sf::Text::getColor () const`

Get the fill color of the text.

Returns

Fill color of the text

See also

[setFillColor](#)

Deprecated:

There is now fill and outline colors instead of a single global color. Use [getOutlineColor\(\)](#) instead.

const Color& sf::Text::getFillColor() const

Get the fill color of the text.

Returns

Fill color of the text

See also

[setFillColor](#)

const Font* sf::Text::getFont() const

Get the text's font.

If the text has no font attached, a NULL pointer is returned. The returned font is read-only, so you cannot modify the font when you get it from this function.

Returns

Pointer to the text's font

See also

[setFont](#)

FloatRect sf::Text::getGlobalBounds () const

Get the global bounding rectangle of the entity.

The returned rectangle is in global coordinates, which means it includes all transformations (translation, rotation, scale, ...) that are applied to the entity. It returns the bounds of the text in the global 2D world's coordinate system.

Returns

Global bounding rectangle of the entity

const Transform& sf::Transformable::getInverseTransform () const

get the inverse of the combined transform of the object

Returns

Inverse of the combined transformations applied to the object

See also

[getTransform](#)

FloatRect sf::Text::getLocalBounds () const

Get the local bounding rectangle of the entity.

The returned rectangle is in local coordinates, which means that (translation, rotation, scale, ...) that are applied to the entity. In other bounds of the entity in the entity's coordinate system.

Returns

Local bounding rectangle of the entity

const **Vector2f&** sf::Transformable::getOrigin () const

get the local origin of the object

Returns

Current origin

See also

[setOrigin](#)

const **Color&** sf::Text::getOutlineColor () const

Get the outline color of the text.

Returns

Outline color of the text

See also

[setOutlineColor](#)

float sf::Text::getOutlineThickness () const

Get the outline thickness of the text.

Returns

Outline thickness of the text, in pixels

See also

[setOutlineThickness](#)

const `Vector2f&` sf::Transformable::getPosition () const

get the position of the object

Returns

Current position

See also

[setPosition](#)

float sf::Transformable::getRotation () const

get the orientation of the object

The rotation is always in the range [0, 360].

Returns

Current rotation, in degrees

See also

[setRotation](#)

const Vector2f& sf::Transformable::getScale() const

get the current scale of the object

Returns

Current scale factors

See also

[setScale](#)

const String& sf::Text::getString() const

Get the text's string.

The returned string is a [sf::String](#), which can automatically be converted
following lines of code are all valid:

```
sf::String    s1 = text.getString();
std::string   s2 = text.getString();
std::wstring  s3 = text.getString();
```

Returns

[Text](#)'s string

See also

[setString](#)

Uint32 sf::Text::getStyle() const

Get the text's style.

Returns

`Text`'s style

See also

`setStyle`

`const Transform& sf::Transformable::getTransform () const`

get the combined transform of the object

Returns

`Transform` combining the position/rotation/scale/origin of the object

See also

`getInverseTransform`

`void sf::Transformable::move (float offsetX, float offsetY)`

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition`, equivalent to the following code:

```
sf::Vector2f pos = object.getPosition();
object.setPosition(pos.x + offsetX, pos.y + offsetY);
```

Parameters

`offsetX` X offset

`offsetY` Y offset

See also

[setPosition](#)

void sf::Transformable::move (const Vector2f & offset)

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition`, equivalent to the following code:

```
object.setPosition(object.getPosition() + offset);
```

Parameters

offset Offset

See also

[setPosition](#)

void sf::Transformable::rotate (float angle)

Rotate the object.

This function adds to the current rotation of the object, unlike `setRotation`, equivalent to the following code:

```
object.setRotation(object.getRotation() + angle);
```

Parameters

angle Angle of rotation, in degrees

```
void sf::Transformable::scale ( float factorX,
                           float factorY
)
```

Scale the object.

This function multiplies the current scale of the object, unlike `setScale` equivalent to the following code:

```
sf::Vector2f scale = object.getScale();
object.setScale(scale.x * factorX, scale.y * factorY);
```

Parameters

factorX Horizontal scale factor
factorY Vertical scale factor

See also

[setScale](#)

```
void sf::Transformable::scale ( const Vector2f & factor )
```

Scale the object.

This function multiplies the current scale of the object, unlike `setScale` equivalent to the following code:

```
sf::Vector2f scale = object.getScale();
object.setScale(scale.x * factor.x, scale.y * factor.y);
```

Parameters

factor Scale factors

See also

[setScale](#)

void sf::Text::setCharacterSize (unsigned int size)

Set the character size.

The default size is 30.

Note that if the used font is a bitmap font, it is not scalable, thus not all to use. This needs to be taken into consideration when setting the character size. When setting the character size of a certain size, make sure the corresponding bitmap font that supports that size exists in the font's character map.

Parameters

size New character size, in pixels

See also

[getCharacterSize](#)

void sf::Text::setColor (const Color & color)

Set the fill color of the text.

By default, the text's fill color is opaque white. Setting the fill color to a color with a low opacity will cause the outline to be displayed in the fill area of the text.

Parameters

color New fill color of the text

See also

[getFillColor](#)

Deprecated:

There is now fill and outline colors instead of a single global color. Use `setOutlineColor()` instead.

`void sf::Text::setFillColor (const Color & color)`

Set the fill color of the text.

By default, the text's fill color is opaque white. Setting the fill color to a transparent color will cause the outline to be displayed in the fill area of the text.

Parameters

color New fill color of the text

See also

`getFillColor`

`void sf::Text::setFont (const Font & font)`

Set the text's font.

The *font* argument refers to a font that must exist as long as the text. It does not store its own copy of the font, but rather keeps a pointer to the one that was passed. If the font is destroyed and the text tries to use it, the behavior is undefined.

Parameters

font New font

See also

`getFont`

```
void sf::Transformable::setOrigin ( float x,  
                                  float y  
    )
```

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transfor

Parameters

- x** X coordinate of the new origin
- y** Y coordinate of the new origin

See also

[getOrigin](#)

```
void sf::Transformable::setOrigin ( const Vector2f & origin )
```

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transfor

Parameters

- origin** New origin

See also

[getOrigin](#)

void sf::Text::setOutlineColor (const Color & color)

Set the outline color of the text.

By default, the text's outline color is opaque black.

Parameters

color New outline color of the text

See also

[getOutlineColor](#)

void sf::Text::setOutlineThickness (float thickness)

Set the thickness of the text's outline.

By default, the outline thickness is 0.

Be aware that using a negative value for the outline thickness will cause

Parameters

thickness New outline thickness, in pixels

See also

[getOutlineThickness](#)

void sf::Transformable::setPosition (float x, float y)

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

x X coordinate of the new position

y Y coordinate of the new position

See also

[move](#), [getPosition](#)

void sf::Transformable::setPosition (const [Vector2f](#) & position)

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

position New position

See also

[move](#), [getPosition](#)

void sf::Transformable::setRotation (float angle)

set the orientation of the object

This function completely overwrites the previous rotation. See the rotate

on the previous rotation instead. The default rotation of a transformable object is 0 degrees.

Parameters

angle New rotation, in degrees

See also

[rotate](#), [getRotation](#)

```
void sf::Transformable::setScale ( float factorX,  
                                 float factorY  
                               )
```

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function for more information. To keep the previous scale instead. The default scale of a transformable object is (1, 1).

Parameters

factorX New horizontal scale factor

factorY New vertical scale factor

See also

[scale](#), [getScale](#)

```
void sf::Transformable::setScale ( const Vector2f & factors )
```

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function for more information. To keep the previous scale instead. The default scale of a transformable object is (1, 1).

Parameters

factors New scale factors

See also

[scale](#), [getScale](#)

void sf::Text::setString (const String & string)

Set the text's string.

The *string* argument is a [sf::String](#), which can automatically be constructed from various types. So, the following calls are all valid:

```
text.setString("hello");
text.setString(L"hello");
text.setString(std::string("hello"));
text.setString(std::wstring(L"hello"));
```

A text's string is empty by default.

Parameters

string New string

See also

[getString](#)

void sf::Text::setStyle (UInt32 style)

Set the text's style.

You can pass a combination of one or more styles, for example [sf::Text::Style::Regular](#). The style is [sf::Text::Regular](#).

Parameters

style New style

See also

[getStyle](#)

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

- [Text.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [Static Public Member Functions](#) | [Static Private Member Functions](#) | [Friends](#) | [L](#)

sf::Texture Class Reference

Graphics module

Image living on the graphics card that can be used for drawing. [More...](#)

```
#include <Texture.hpp>
```

Inheritance diagram for sf::Texture:



Public Types

enum **CoordinateType** { **Normalized**, **Pixels** }

Types of texture coordinates that can be used for rendering. More

Public Member Functions

`Texture ()`

Default constructor. [More...](#)

`Texture (const Texture ©)`

Copy constructor. [More...](#)

`~Texture ()`

Destructor. [More...](#)

`bool create (unsigned int width, unsigned int height)`

Create the texture. [More...](#)

`bool loadFromFile (const std::string &filename, const IntRect &area)`

Load the texture from a file on disk. [More...](#)

`bool loadFromMemory (const void *data, std::size_t size, const IntRect &area)`

Load the texture from a file in memory. [More...](#)

`bool loadFromStream (InputStream &stream, const IntRect &area)`

Load the texture from a custom stream. [More...](#)

`bool loadFromImage (const Image &image, const IntRect &area)`

Load the texture from an image. [More...](#)

`Vector2u getSize () const`

Return the size of the texture. [More...](#)

`Image copyToImage () const`

Copy the texture pixels to an image. [More...](#)

`void update (const Uint8 *pixels)`

Update the whole texture from an array of pixels. [More...](#)

void `update (const Uint8 *pixels, unsigned int width, unsigned int height, int y)`

Update a part of the texture from an array of pixels. [More...](#)

void `update (const Image &image)`

Update the texture from an image. [More...](#)

void `update (const Image &image, unsigned int x, unsigned int y)`

Update a part of the texture from an image. [More...](#)

void `update (const Window &window)`

Update the texture from the contents of a window. [More...](#)

void `update (const Window &window, unsigned int x, unsigned int y)`

Update a part of the texture from the contents of a window.

void `setSmooth (bool smooth)`

Enable or disable the smooth filter. [More...](#)

bool `isSmooth () const`

Tell whether the smooth filter is enabled or not. [More...](#)

void `setSrgb (bool sRgb)`

Enable or disable conversion from sRGB. [More...](#)

bool `isSrgb () const`

Tell whether the texture source is converted from sRGB or not. [More...](#)

void `setRepeated (bool repeated)`

Enable or disable repeating. [More...](#)

bool `isRepeated () const`

Tell whether the texture is repeated or not. [More...](#)

`bool generateMipmap ()`

Generate a mipmap using the current texture data. [More...](#)

`Texture & operator= (const Texture &right)`

Overload of assignment operator. [More...](#)

`unsigned int getNativeHandle () const`

Get the underlying OpenGL handle of the texture. [More...](#)

Static Public Member Functions

```
static void bind (const Texture *texture, CoordinateType coordir  
Bind a texture for rendering. More...
```

```
static unsigned int getMaximumSize ()  
Get the maximum texture size allowed. More...
```

Static Private Member Functions

```
static void ensureGLContext ()
```

Empty function for ABI compatibility, use acquireTransientContext()

Friends

class **RenderTexture**

class **RenderTarget**

Detailed Description

Image living on the graphics card that can be used for drawing.

`sf::Texture` stores pixels that can be drawn, with a sprite for example.

A texture lives in the graphics card memory, therefore it is very fast to draw. You can copy a render target to a texture (the graphics card can access both directly).

Being stored in the graphics card memory has some drawbacks. A texture cannot be modified directly. If you want to change the pixels of a texture, do it as a `sf::Image`, you need to prepare the pixels first and then upload them to the texture via a `sf::Texture::update` operation (see `Texture::update`).

`sf::Texture` makes it easy to convert from/to `sf::Image`, but keep in mind that these conversions involve data transfer between the graphics card and the central memory, therefore they are slower than direct access to the texture.

A texture can be loaded from an image, but also directly from a file. Several file formats are supported. Texture loading shortcuts are defined so that you don't need an image first for the most part. If you want to modify the pixels of a texture, do it as a `sf::Image`, do whatever you need with the pixels, and then call `Texture::update`.

Since they live in the graphics card memory, the pixels of a texture cannot be modified individually. They can only be modified in blocks (for example for pixel-perfect collisions), it is recommended to store the collision information in a separate array of booleans.

Like `sf::Image`, `sf::Texture` can handle a unique internal representation of colors. This means that a pixel must be composed of 8 bits red, green, blue and 8 bits alpha. It is represented by the `sf::Color`.

Usage example:

```

// This example shows the most common use of sf::Texture:
// drawing a sprite

// Load a texture from a file
sf::Texture texture;
if (!texture.loadFromFile("texture.png"))
    return -1;

// Assign it to a sprite
sf::Sprite sprite;
sprite.setTexture(texture);

// Draw the textured sprite
window.draw(sprite);

// This example shows another common use of sf::Texture:
// streaming real-time data, like video frames

// Create an empty texture
sf::Texture texture;
if (!texture.create(640, 480))
    return -1;

// Create a sprite that will display the texture
sf::Sprite sprite(texture);

while (...) // the main loop
{
    ...

    // update the texture
    sf::Uint8* pixels = ...; // get a fresh chunk of pixels (the next 1
    texture.update(pixels);

    // draw it
    window.draw(sprite);

    ...
}

}

```

Like `sf::Shader` that can be used as a raw OpenGL shader, `sf::Texture` can be used as a raw OpenGL texture for custom OpenGL geometry.

```

sf::Texture::bind(&texture);
... render OpenGL geometry ...
sf::Texture::bind(NULL);

```

See also

`sf::Sprite`, `sf::Image`, `sf::RenderTexture`

Definition at line 47 of file `Texture.hpp`.

Member Enumeration Documentation

enum sf::Texture::CoordinateType

Types of texture coordinates that can be used for rendering.

Enumerator

Normalized

Texture coordinates in range [0 .. 1].

Pixels

Texture coordinates in range [0 .. size].

Definition at line 55 of file [Texture.hpp](#).

Constructor & Destructor Documentation

sf::Texture::Texture()

Default constructor.

Creates an empty texture.

sf::Texture::Texture(const Texture & copy)

Copy constructor.

Parameters

copy instance to copy

sf::Texture::~Texture()

Destructor.

Member Function Documentation

```
static void sf::Texture::bind ( const Texture * texture,
                               CoordinateType coordinateType = Normal )

```

Bind a texture for rendering.

This function is not part of the graphics API, it mustn't be used when drawing with OpenGL. It is used only if you mix `sf::Texture` with OpenGL code.

```
sf::Texture t1, t2;
...
sf::Texture::bind(&t1);
// draw OpenGL stuff that use t1...
sf::Texture::bind(&t2);
// draw OpenGL stuff that use t2...
sf::Texture::bind(NULL);
// draw OpenGL stuff that use no texture...
```

The `coordinateType` argument controls how texture coordinates will be handled (by default, they must be in range [0 .. 1], which is the default way of handling them in OpenGL). If Pixels, they must be given in pixels (range [0 .. size]). This makes the definition of texture coordinates easier: in the graphics classes of SFML, it makes the definition of texture coordinates easier for the API, users don't need to compute normalized values.

Parameters

texture Pointer to the texture to bind, can be null to use no texture
coordinateType Type of texture coordinates to use

Image `sf::Texture::copyToImage() const`

Copy the texture pixels to an image.

This function performs a slow operation that downloads the texture's pixels and copies them to a new image, potentially applying transformations to pixels (e.g. padded or flipped).

Returns

`Image` containing the texture's pixels

See also

`loadFromImage`

```
bool sf::Texture::create( unsigned int width,  
                           unsigned int height  
                         )
```

Create the texture.

If this function fails, the texture is left unchanged.

Parameters

width Width of the texture

height Height of the texture

Returns

True if creation was successful

```
bool sf::Texture::generateMipmap( )
```

Generate a mipmap using the current texture data.

Mipmaps are pre-computed chains of optimized textures. Each level of the mipmap is generated by halving each of the previous level's dimensions. This is done until the base level image is reached. The textures generated in this process may make use of more advanced filtering techniques to maintain visual quality of textures when they are applied to objects much smaller than their original size. Mipmapping is also known as minification. Because fewer texels (texture elements) have to be sampled when rendering small objects, the usage of mipmaps can also improve rendering performance in certain scenarios.

Mipmap generation relies on the necessary OpenGL extension being available. If mipmap generation fails due to another reason, this function will return false. Mipmaps are automatically generated until the next time the base level image is modified. To regenerate the mipmap chain, the `generateMipmap` function has to be called again to regenerate it.

Returns

True if mipmap generation was successful, false if unsuccessful

static unsigned int sf::Texture::getMaximumSize ()

Get the maximum texture size allowed.

This maximum size is defined by the graphics driver. You can expect a value between 256 and 8192 pixels, depending on the capabilities of the graphics card, and up to 8192 pixels or more for newer hardware.

Returns

Maximum size allowed for textures, in pixels

unsigned int sf::Texture::getNativeHandle () const

Get the underlying OpenGL handle of the texture.

You shouldn't need to use this function, unless you have very specific requirements. It's mainly provided for compatibility with OpenGL, which doesn't support, or implement a temporary workaround until a bug is fixed.

Returns

OpenGL handle of the texture or 0 if not yet created

Vector2u sf::Texture::getSize() const

Return the size of the texture.

Returns

Size in pixels

bool sf::Texture::isRepeated() const

Tell whether the texture is repeated or not.

Returns

True if repeat mode is enabled, false if it is disabled

See also

[setRepeated](#)

bool sf::Texture::isSmooth() const

Tell whether the smooth filter is enabled or not.

Returns

True if smoothing is enabled, false if it is disabled

See also

[setSmooth](#)

bool sf::Texture::isSrgb() const

Tell whether the texture source is converted from sRGB or not.

Returns

True if the texture source is converted from sRGB, false if not

See also

[setSrgb](#)

bool sf::Texture::loadFromFile (const std::string & filename, const IntRect & area = IntRect ())

Load the texture from a file on disk.

This function is a shortcut for the following code:

```
sf::Image image;
image.loadFromFile(filename);
texture.loadFromImage(image, area);
```

The `area` argument can be used to load only a sub-rectangle of the whole image then leave the default value (which is an empty `IntRect`). If the `area` of the image, it is adjusted to fit the image size.

The maximum size for a texture depends on the graphics driver's `getMaximumSize` function.

If this function fails, the texture is left unchanged.

Parameters

filename Path of the image file to load

area Area of the image to load

Returns

True if loading was successful

See also

[loadFromMemory](#), [loadFromStream](#), [loadFromImage](#)

```
bool sf::Texture::loadFromImage( const Image & image,
                                const IntRect & area = IntRect() )
```

Load the texture from an image.

The *area* argument can be used to load only a sub-rectangle of the whole image then leave the default value (which is an empty IntRect). If the *area* is outside the boundaries of the image, it is adjusted to fit the image size.

The maximum size for a texture depends on the graphics driver's [getMaximumSize](#) function.

If this function fails, the texture is left unchanged.

Parameters

image [Image](#) to load into the texture

area Area of the image to load

Returns

True if loading was successful

See also

[loadFromFile](#), [loadFromMemory](#)

```
bool sf::Texture::loadFromMemory ( const void *      data,
                                  std::size_t        size,
                                  const IntRect & area = IntRect()
)
```

Load the texture from a file in memory.

This function is a shortcut for the following code:

```
sf::Image image;
image.loadFromMemory(data, size);
texture.loadFromImage(image, area);
```

The `area` argument can be used to load only a sub-rectangle of the whole image then leave the default value (which is an empty `IntRect`). If the `area` is outside of the image, it is adjusted to fit the image size.

The maximum size for a texture depends on the graphics driver and the `getMaximumSize` function.

If this function fails, the texture is left unchanged.

Parameters

data Pointer to the file data in memory

size Size of the data to load, in bytes

area Area of the image to load

Returns

True if loading was successful

See also

[loadFromFile](#), [loadFromStream](#), [loadFromImage](#)

```
bool sf::Texture::loadFromStream ( InputStream & stream,
                                    const IntRect & area = IntRect () )
```

Load the texture from a custom stream.

This function is a shortcut for the following code:

```
sf::Image image;
image.loadFromStream(stream);
texture.loadFromImage(image, area);
```

The `area` argument can be used to load only a sub-rectangle of the whole image then leave the default value (which is an empty `IntRect`). If the `area` of the image, it is adjusted to fit the image size.

The maximum size for a texture depends on the graphics driver's `getMaximumSize` function.

If this function fails, the texture is left unchanged.

Parameters

stream Source stream to read from

area Area of the image to load

Returns

True if loading was successful

See also

`loadFromFile`, `loadFromMemory`, `loadFromImage`

Texture& sf::Texture::operator= (const Texture & right)

Overload of assignment operator.

Parameters

right Instance to assign

Returns

Reference to self

void sf::Texture::setRepeated (bool repeated)

Enable or disable repeating.

Repeating is involved when using texture coordinates outside the texture's boundaries. In this case, if repeat mode is enabled, the whole texture will be repeated until it reaches the coordinate (for example, if the X texture coordinate is 3 * width, it will be repeated 3 times). If repeat mode is disabled, the "extra space" will instead be filled with white pixels. On very old graphics cards, white pixels may appear when the texture is repeated. This mode can be used reliably only if the texture has power-of-two dimensions. Repeating is disabled by default.

Parameters

repeated True to repeat the texture, false to disable repeating

See also

`isRepeated`

void sf::Texture::setSmooth (bool smooth)

Enable or disable the smooth filter.

When the filter is activated, the texture appears smoother so that pixels you want the texture to look exactly the same as its source file, you should note that it will take more memory and processing power. The smooth filter is disabled by default.

Parameters

smooth True to enable smoothing, false to disable it

See also

[isSmooth](#)

void sf::Texture::setSrgb (bool sRgb)

Enable or disable conversion from sRGB.

When providing texture data from an image file or memory, it can either be provided in linear color space or in sRGB color space. Most digital images account for gamma correction and are therefore "corrected" back to linear color space before being processed. This function allows you to enable or disable this conversion. When the rendered image gets output to the final framebuffer, it gets converted back to sRGB.

After enabling or disabling sRGB conversion, make sure to reload the texture to take effect.

This option is only useful in conjunction with an sRGB capable framebuffer or window creation.

Parameters

sRgb True to enable sRGB conversion, false to disable it

See also

[isSrgb](#)

void sf::Texture::update (const Uint8 * pixels)

Update the whole texture from an array of pixels.

The *pixel* array is assumed to have the same size as the *area* rectangle pixels.

No additional check is performed on the size of the pixel array, passing undefined behavior.

This function does nothing if *pixels* is null or if the texture was not previously created.

Parameters

pixels Array of pixels to copy to the texture

void sf::Texture::update (const Uint8 * pixels, unsigned int width, unsigned int height, unsigned int x, unsigned int y)

Update a part of the texture from an array of pixels.

The size of the *pixel* array must match the *width* and *height* arguments, and its pixels.

No additional check is performed on the size of the pixel array or the

passing invalid arguments will lead to an undefined behavior.

This function does nothing if *pixels* is null or if the texture was not previously created.

Parameters

pixels Array of pixels to copy to the texture

width Width of the pixel region contained in *pixels*

height Height of the pixel region contained in *pixels*

x X offset in the texture where to copy the source pixels

y Y offset in the texture where to copy the source pixels

`void sf::Texture::update (const Image & image)`

Update the texture from an image.

Although the source image can be smaller than the texture, this function copies the whole image to the whole texture. The other overload, which has (x, y) additional arguments, is used for updating a sub-area of the texture.

No additional check is performed on the size of the image, passing an invalid image will lead to an undefined behavior.

This function does nothing if the texture was not previously created.

Parameters

image Image to copy to the texture

`void sf::Texture::update (const Image & image, unsigned int x, unsigned int y)`

Update a part of the texture from an image.

No additional check is performed on the size of the image, passing an invalid offset will lead to an undefined behavior.

This function does nothing if the texture was not previously created.

Parameters

image `Image` to copy to the texture

x X offset in the texture where to copy the source image

y Y offset in the texture where to copy the source image

`void sf::Texture::update (const Window & window)`

Update the texture from the contents of a window.

Although the source window can be smaller than the texture, this function updates the whole texture. The other overload, which has (x, y) additional arguments, is for updating a sub-area of the texture.

No additional check is performed on the size of the window, passing a window with an invalid size will lead to an undefined behavior.

This function does nothing if either the texture or the window was not previously created.

Parameters

window `Window` to copy to the texture

`void sf::Texture::update (const Window & window,`

unsigned int **x**,

unsigned int **y**

)

Update a part of the texture from the contents of a window.

No additional check is performed on the size of the window, passing a size and offset will lead to an undefined behavior.

This function does nothing if either the texture or the window was not pre-

Parameters

window [Window](#) to copy to the texture

x X offset in the texture where to copy the source window

y Y offset in the texture where to copy the source window

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

- [Texture.hpp](#)

SFML is licensed under the terms and conditions of the [zlib/png license](#).

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#).

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Static Public Attributes](#) | [Related Functions](#) | [List of all members](#)

sf::Transform Class Reference

[Graphics module](#)

Define a 3x3 transform matrix. [More...](#)

```
#include <Transform.hpp>
```

Public Member Functions

`Transform ()`

Default constructor. [More...](#)

`Transform (float a00, float a01, float a02, float a10, float a11, float a22)`

Construct a transform from a 3x3 matrix. [More...](#)

`const float * getMatrix () const`

Return the transform as a 4x4 matrix. [More...](#)

`Transform getInverse () const`

Return the inverse of the transform. [More...](#)

`Vector2f transformPoint (float x, float y) const`

Transform a 2D point. [More...](#)

`Vector2f transformPoint (const Vector2f &point) const`

Transform a 2D point. [More...](#)

`FloatRect transformRect (const FloatRect &rectangle) const`

Transform a rectangle. [More...](#)

`Transform & combine (const Transform &transform)`

Combine the current transform with another one. [More...](#)

`Transform & translate (float x, float y)`

Combine the current transform with a translation. [More...](#)

`Transform & translate (const Vector2f &offset)`

Combine the current transform with a translation. [More...](#)

Transform & rotate (float angle)

Combine the current transform with a rotation. More...

Transform & rotate (float angle, float centerX, float centerY)

Combine the current transform with a rotation. More...

Transform & rotate (float angle, const Vector2f ¢er)

Combine the current transform with a rotation. More...

Transform & scale (float scaleX, float scaleY)

Combine the current transform with a scaling. More...

Transform & scale (float scaleX, float scaleY, float centerX, float centerY)

Combine the current transform with a scaling. More...

Transform & scale (const Vector2f &factors)

Combine the current transform with a scaling. More...

Transform & scale (const Vector2f &factors, const Vector2f ¢er)

Combine the current transform with a scaling. More...

Static Public Attributes

static const [Transform](#) [Identity](#)

The identity transform (does nothing) [More...](#)

Related Functions

(Note that these are not member functions.)

`Transform operator* (const Transform &left, const Transform &right)`
Overload of binary operator * to combine two transforms. [More...](#)

`Transform & operator*=(Transform &left, const Transform &right)`
Overload of binary operator *= to combine two transforms.

`Vector2f operator* (const Transform &left, const Vector2f &right)`
Overload of binary operator * to transform a point. [More...](#)

Detailed Description

Define a 3x3 transform matrix.

A sf::Transform specifies how to translate, rotate, scale, shear, project, wh

In mathematical terms, it defines how to transform a coordinate system int

For example, if you apply a rotation transform to a sprite, the result will
that is transformed by this rotation transform will be rotated the same way,

Transforms are typically used for drawing. But they can also be used for
transform points between the local and global coordinate systems of an ei

Example:

```
// define a translation transform
sf::Transform translation;
translation.translate(20, 50);

// define a rotation transform
sf::Transform rotation;
rotation.rotate(45);

// combine them
sf::Transform transform = translation * rotation;

// use the result to transform stuff...
sf::Vector2f point = transform.transformPoint(10, 20);
sf::FloatRect rect = transform.transformRect(sf::FloatRect(0, 0, 10, 10));
```

See also

[sf::Transformable](#), [sf::RenderStates](#)

Definition at line 42 of file [Transform.hpp](#).

Constructor & Destructor Documentation

sf::Transform::Transform()

Default constructor.

Creates an identity transform (a transform that does nothing).

```
sf::Transform::Transform( float a00,
                      float a01,
                      float a02,
                      float a10,
                      float a11,
                      float a12,
                      float a20,
                      float a21,
                      float a22
)
```

Construct a transform from a 3x3 matrix.

Parameters

- a00** Element (0, 0) of the matrix
- a01** Element (0, 1) of the matrix
- a02** Element (0, 2) of the matrix
- a10** Element (1, 0) of the matrix
- a11** Element (1, 1) of the matrix

a12 Element (1, 2) of the matrix

a20 Element (2, 0) of the matrix

a21 Element (2, 1) of the matrix

a22 Element (2, 2) of the matrix

Member Function Documentation

Transform& sf::Transform::combine (const Transform & transform

Combine the current transform with another one.

The result is a transform that is equivalent to applying `*this` followed by equivalent to a matrix multiplication.

Parameters

transform Transform to combine with this transform

Returns

Reference to `*this`

Transform sf::Transform::getInverse () const

Return the inverse of the transform.

If the inverse cannot be computed, an identity transform is returned.

Returns

A new transform which is the inverse of self

const float* sf::Transform::getMatrix () const

Return the transform as a 4x4 matrix.

This function returns a pointer to an array of 16 floats containing the transform which is directly compatible with OpenGL functions.

```
sf::Transform transform = ...;
glLoadMatrixf(transform.getMatrix());
```

Returns

Pointer to a 4x4 matrix

Transform& sf::Transform::rotate (float angle)

Combine the current transform with a rotation.

This function returns a reference to *this, so that calls can be chained.

```
sf::Transform transform;
transform.rotate(90).translate(50, 20);
```

Parameters

angle Rotation angle, in degrees

Returns

Reference to *this

See also

translate, scale

Transform& sf::Transform::rotate (float angle, float centerX,

```
float centerY  
)
```

Combine the current transform with a rotation.

The center of rotation is provided for convenience as a second argument around arbitrary points more easily (and efficiently) than `center).rotate(angle).translate(center)`.

This function returns a reference to `*this`, so that calls can be chained.

```
sf::Transform transform;  
transform.rotate(90, 8, 3).translate(50, 20);
```

Parameters

- angle** Rotation angle, in degrees
- centerX** X coordinate of the center of rotation
- centerY** Y coordinate of the center of rotation

Returns

Reference to `*this`

See also

`translate`, `scale`

```
Transform& sf::Transform::rotate ( float angle,  
                                const Vector2f & center  
)
```

Combine the current transform with a rotation.

The center of rotation is provided for convenience as a second argument around arbitrary points more easily (and efficiently) than `center).rotate(angle).translate(center)`.

`center).rotate(angle).translate(center).`

This function returns a reference to `*this`, so that calls can be chained.

```
sf::Transform transform;
transform.rotate(90, sf::Vector2f(8, 3)).translate(sf::Vector2f(50, 2))
```

Parameters

angle Rotation angle, in degrees

center Center of rotation

Returns

Reference to `*this`

See also

`translate`, `scale`

**Transform& sf::Transform::scale (float scaleX,
float scaleY
)**

Combine the current transform with a scaling.

This function returns a reference to `*this`, so that calls can be chained.

```
sf::Transform transform;
transform.scale(2, 1).rotate(45);
```

Parameters

scaleX Scaling factor on the X axis

scaleY Scaling factor on the Y axis

Returns

Reference to `*this`

See also

`translate`, `rotate`

```
Transform& sf::Transform::scale ( float scaleX,  
                                float scaleY,  
                                float centerX,  
                                float centerY  
                            )
```

Combine the current transform with a scaling.

The center of scaling is provided for convenience as a second argument around arbitrary points more easily (and efficiently) than `center.scale(factors).translate(center)`.

This function returns a reference to `*this`, so that calls can be chained.

```
sf::Transform transform;  
transform.scale(2, 1, 8, 3).rotate(45);
```

Parameters

`scaleX` Scaling factor on X axis

`scaleY` Scaling factor on Y axis

`centerX` X coordinate of the center of scaling

`centerY` Y coordinate of the center of scaling

Returns

Reference to `*this`

See also

`translate`, `rotate`

Transform& sf::Transform::scale (const Vector2f & factors)

Combine the current transform with a scaling.

This function returns a reference to *this, so that calls can be chained.

```
sf::Transform transform;
transform.scale(sf::Vector2f(2, 1)).rotate(45);
```

Parameters

factors Scaling factors

Returns

Reference to *this

See also

translate, rotate

Transform& sf::Transform::scale (const Vector2f & factors, const Vector2f & center)

Combine the current transform with a scaling.

The center of scaling is provided for convenience as a second argument around arbitrary points more easily (and efficiently) than center).scale(factors).translate(center).

This function returns a reference to *this, so that calls can be chained.

```
sf::Transform transform;
```

```
transform.scale(sf::Vector2f(2, 1), sf::Vector2f(8, 3)).rotate(45);
```

Parameters

factors Scaling factors
center Center of scaling

Returns

Reference to *this

See also

translate, rotate

```
Vector2f sf::Transform::transformPoint ( float x,  
                                         float y  
                                       ) const
```

Transform a 2D point.

Parameters

x X coordinate of the point to transform
y Y coordinate of the point to transform

Returns

Transformed point

```
Vector2f sf::Transform::transformPoint ( const Vector2f & point ) co
```

Transform a 2D point.

Parameters

point Point to transform

Returns

Transformed point

FloatRect sf::Transform::transformRect (const FloatRect & rectangle)

Transform a rectangle.

Since SFML doesn't provide support for oriented rectangles, the result is an aligned rectangle. Which means that if the transform contains a rotation, the transformed rectangle is returned.

Parameters

rectangle Rectangle to transform

Returns

Transformed rectangle

Transform& sf::Transform::translate (float x, float y)

Combine the current transform with a translation.

This function returns a reference to *this, so that calls can be chained.

```
sf::Transform transform;
transform.translate(100, 200).rotate(45);
```

Parameters

x Offset to apply on X axis

y Offset to apply on Y axis

Returns

Reference to *this

See also

[rotate](#), [scale](#)

Transform& sf::Transform::translate (const Vector2f & offset)

Combine the current transform with a translation.

This function returns a reference to *this, so that calls can be chained.

```
sf::Transform transform;
transform.translate(sf::Vector2f(100, 200)).rotate(45);
```

Parameters

offset Translation offset to apply

Returns

Reference to *this

See also

[rotate](#), [scale](#)

Friends And Related Function Documentation

```
Transform operator* ( const Transform & left,  
                     const Transform & right  
                 )
```

Overload of binary operator * to combine two transforms.

This call is equivalent to calling `Transform(left).combine(right)`.

Parameters

left Left operand (the first transform)
right Right operand (the second transform)

Returns

New combined transform

```
Vector2f operator* ( const Transform & left,  
                     const Vector2f & right  
                 )
```

Overload of binary operator * to transform a point.

This call is equivalent to calling `left.transformPoint(right)`.

Parameters

left Left operand (the transform)
right Right operand (the point to transform)

Returns

New transformed point

```
Transform & operator*=( Transform & left,  
                           const Transform & right  
                         )
```

Overload of binary operator *= to combine two transforms.

This call is equivalent to calling left.combine(right).

Parameters

left Left operand (the first transform)

right Right operand (the second transform)

Returns

The combined transform

Member Data Documentation

const Transform sf::Transform::Identity

The identity transform (does nothing)

Definition at line 354 of file [Transform.hpp](#).

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

- [Transform.hpp](#)
-

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Member Functions](#) | [List of all members](#)

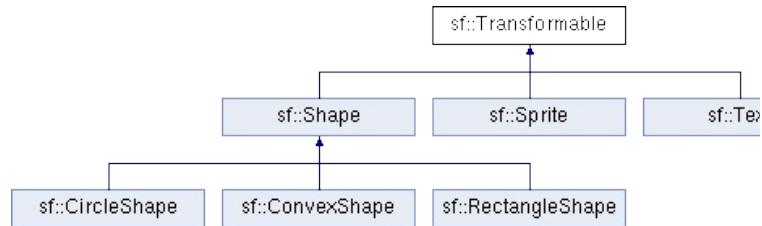
sf::Transformable Class Reference

Graphics module

Decomposed transform defined by a position, a rotation and a scale. More...

```
#include <Transformable.hpp>
```

Inheritance diagram for sf::Transformable:



Public Member Functions

`Transformable ()`
Default constructor. [More...](#)

`virtual ~Transformable ()`
Virtual destructor. [More...](#)

`void setPosition (float x, float y)`
set the position of the object [More...](#)

`void setPosition (const Vector2f &position)`
set the position of the object [More...](#)

`void setRotation (float angle)`
set the orientation of the object [More...](#)

`void setScale (float factorX, float factorY)`
set the scale factors of the object [More...](#)

`void setScale (const Vector2f &factors)`
set the scale factors of the object [More...](#)

`void setOrigin (float x, float y)`
set the local origin of the object [More...](#)

`void setOrigin (const Vector2f &origin)`
set the local origin of the object [More...](#)

`const Vector2f & getPosition () const`
get the position of the object [More...](#)

`float getRotation () const`

get the orientation of the object [More...](#)

const Vector2f & [getScale \(\) const](#)
get the current scale of the object [More...](#)

const Vector2f & [getOrigin \(\) const](#)
get the local origin of the object [More...](#)

void [move \(float offsetX, float offsetY\)](#)
Move the object by a given offset. [More...](#)

void [move \(const Vector2f &offset\)](#)
Move the object by a given offset. [More...](#)

void [rotate \(float angle\)](#)
Rotate the object. [More...](#)

void [scale \(float factorX, float factorY\)](#)
Scale the object. [More...](#)

void [scale \(const Vector2f &factor\)](#)
Scale the object. [More...](#)

const Transform & [getTransform \(\) const](#)
get the combined transform of the object [More...](#)

const Transform & [getInverseTransform \(\) const](#)
get the inverse of the combined transform of the obj

Detailed Description

Decomposed transform defined by a position, a rotation and a scale.

This class is provided for convenience, on top of `sf::Transform`.

`sf::Transform`, as a low-level class, offers a great level of flexibility but is hard to manage. Indeed, one can easily combine any kind of operation, such as a translation followed by a scaling, but once the result transform is built, there's no way to change only the rotation without modifying the translation and scaling. This requires recomputing the final transform, which means that you need to retrieve the initial transformation, change only the rotation without modifying the translation and scaling, and then recomputed, which means that you need to retrieve the initial transformation again and combine them the same way you did before updating the rotation. This is why it's better to use `sf::Transformable` instead of `sf::Transform`.

That's exactly what `sf::Transformable` was written for: it hides these various operations behind an easy to use interface. You can set or get any of the individual components without worrying about the others. It also provides the composed transform (as a `sf::Transform`) for you.

In addition to the position, rotation and scale, `sf::Transformable` provides the origin of the local coordinate system. This origin represents the local origin of the three other components. Let's take an example. By default, the sprite is positioned/rotated/scaled relatively to its top-left corner (0, 0). But if we change the origin to be (5, 5), the sprite will be positioned around (5, 5) instead of (0, 0). And if we set the origin to (10, 10), it will be transformed around it instead of (0, 0).

To keep the `sf::Transformable` class simple, there's only one origin for all components. If you want to position the sprite relatively to its top-left corner while rotating it around it, or if you want to rotate it around its center instead of its top-left corner, use `sf::Transform` directly.

`sf::Transformable` can be used as a base class. It is often combined with other classes, such as `sf::Image`, `sf::Text` and `sf::Shape`. These classes inherit from `sf::Transformable` and do most of the work for you.

```

class MyEntity : public sf::Transformable, public sf::Drawable
{
    virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
    {
        states.transform *= getTransform();
        target.draw(..., states);
    }
};

MyEntity entity;
entity.setPosition(10, 20);
entity.setRotation(45);
window.draw(entity);

```

It can also be used as a member, if you don't want to use its API direct functions, or you have different naming conventions for example).

```

class MyEntity
{
public:
    void SetPosition(const MyVector& v)
    {
        myTransform.setPosition(v.x(), v.y());
    }

    void Draw(sf::RenderTarget& target) const
    {
        target.draw(..., myTransform.getTransform());
    }

private:
    sf::Transformable myTransform;
};

```

A note on coordinates and undistorted rendering:

By default, SFML (or more exactly, OpenGL) may interpolate drawable when rendering. While this allows transitions like slow movements or rotations to lead to unwanted results in some cases, for example blurred or distorted images. To render a `sf::Drawable` object pixel-perfectly, make sure the involved coordinates align with the window's texels (pixels in the texture). More specifically, this means:

- The object's position, origin and scale have no fractional part
- The object's and the view's rotation are a multiple of 90 degrees

- The view's center and size have no fractional part

See also

[sf::Transform](#)

Definition at line [41](#) of file [Transformable.hpp](#).

Constructor & Destructor Documentation

sf::Transformable::Transformable()

Default constructor.

virtual sf::Transformable::~Transformable()

Virtual destructor.

Member Function Documentation

const Transform& sf::Transformable::getInverseTransform () const

get the inverse of the combined transform of the object

Returns

Inverse of the combined transformations applied to the object

See also

[getTransform](#)

const Vector2f& sf::Transformable::getOrigin () const

get the local origin of the object

Returns

Current origin

See also

[setOrigin](#)

const Vector2f& sf::Transformable::getPosition () const

get the position of the object

Returns

Current position

See also

[setPosition](#)

float sf::Transformable::getRotation() const

get the orientation of the object

The rotation is always in the range [0, 360].

Returns

Current rotation, in degrees

See also

[setRotation](#)

const Vector2f& sf::Transformable::getScale() const

get the current scale of the object

Returns

Current scale factors

See also

[setScale](#)

const Transform& sf::Transformable::getTransform() const

get the combined transform of the object

Returns

`Transform` combining the position/rotation/scale/origin of the object

See also

`getInverseTransform`

```
void sf::Transformable::move ( float offsetX,  
                             float offsetY  
                           )
```

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition`, equivalent to the following code:

```
sf::Vector2f pos = object.getPosition();  
object.setPosition(pos.x + offsetX, pos.y + offsetY);
```

Parameters

`offsetX` X offset
`offsetY` Y offset

See also

`setPosition`

```
void sf::Transformable::move ( const Vector2f & offset )
```

Move the object by a given offset.

This function adds to the current position of the object, unlike `setPosition` equivalent to the following code:

```
object.setPosition(object.getPosition() + offset);
```

Parameters

offset Offset

See also

[setPosition](#)

void sf::Transformable::rotate (float angle)

Rotate the object.

This function adds to the current rotation of the object, unlike `setRotation` equivalent to the following code:

```
object.setRotation(object.getRotation() + angle);
```

Parameters

angle Angle of rotation, in degrees

void sf::Transformable::scale (float factorX, float factorY)

Scale the object.

This function multiplies the current scale of the object, unlike `setScale` equivalent to the following code:

```
sf::Vector2f scale = object.getScale();
object.setScale(scale.x * factorX, scale.y * factorY);
```

Parameters

factorX Horizontal scale factor

factorY Vertical scale factor

See also

[setScale](#)

void sf::Transformable::scale (const Vector2f & factor)

Scale the object.

This function multiplies the current scale of the object, unlike `setScale`, equivalent to the following code:

```
sf::Vector2f scale = object.getScale();
object.setScale(scale.x * factor.x, scale.y * factor.y);
```

Parameters

factor Scale factors

See also

[setScale](#)

void sf::Transformable::setOrigin (float x, float y)

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transfor

Parameters

x X coordinate of the new origin

y Y coordinate of the new origin

See also

[getOrigin](#)

void sf::Transformable::setOrigin (const Vector2f & origin)

set the local origin of the object

The origin of an object defines the center point for all transformations: coordinates of this point must be relative to the top-left corner transformations (position, scale, rotation). The default origin of a transfor

Parameters

origin New origin

See also

[getOrigin](#)

void sf::Transformable::setPosition (float x, float y)

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

- x** X coordinate of the new position
- y** Y coordinate of the new position

See also

[move](#), [getPosition](#)

void sf::Transformable::setPosition (const Vector2f & position)

set the position of the object

This function completely overwrites the previous position. See the move on the previous position instead. The default position of a transformable

Parameters

- position** New position

See also

[move](#), [getPosition](#)

void sf::Transformable::setRotation (float angle)

set the orientation of the object

This function completely overwrites the previous rotation. See the rotate

on the previous rotation instead. The default rotation of a transformable object is 0 degrees.

Parameters

angle New rotation, in degrees

See also

[rotate](#), [getRotation](#)

```
void sf::Transformable::setScale ( float factorX,  
                                 float factorY  
                               )
```

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function for more information. To keep the previous scale instead. The default scale of a transformable object is (1, 1).

Parameters

factorX New horizontal scale factor

factorY New vertical scale factor

See also

[scale](#), [getScale](#)

```
void sf::Transformable::setScale ( const Vector2f & factors )
```

set the scale factors of the object

This function completely overwrites the previous scale. See the scale function for more information. To keep the previous scale instead. The default scale of a transformable object is (1, 1).

Parameters

factors New scale factors

See also

[scale](#), [getScale](#)

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

- [Transformable.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Public Attributes](#) | [List of all members](#)

sf::Vertex Class Reference

[Graphics module](#)

Define a point with color and texture coordinates. More...

```
#include <Vertex.hpp>
```

Public Member Functions

`Vertex ()`

Default constructor. [More...](#)

`Vertex (const Vector2f &thePosition)`

Construct the vertex from its position. [More...](#)

`Vertex (const Vector2f &thePosition, const Color &theColor)`

Construct the vertex from its position and color. [More...](#)

`Vertex (const Vector2f &thePosition, const Vector2f &theTexCoords)`

Construct the vertex from its position and texture coordinates. [More...](#)

`Vertex (const Vector2f &thePosition, const Color &theColor, const Vec`

Construct the vertex from its position, color and texture coordinates. [M](#)

Public Attributes

Vector2f **position**

2D position of the vertex [More...](#)

Color **color**

Color of the vertex. [More...](#)

Vector2f **texCoords**

Coordinates of the texture's pixel to map to the vertex. [More...](#)

Detailed Description

Define a point with color and texture coordinates.

A vertex is an improved point.

It has a position and other extra attributes that will be used for drawing: in and a pair of texture coordinates.

The vertex is the building block of drawing. Everything which is visible on are grouped as 2D primitives (triangles, quads, ...), and these primitives complex 2D entities such as sprites, texts, etc.

If you use the graphical entities of SFML (sprite, text, shape) you won't have to do all the drawing work yourself. But if you want to define your own 2D entities, such as tiled maps or particle systems, vertices will allow you to get maximum performances.

Example:

```
// define a 100x100 square, red, with a 10x10 texture mapped on it
sf::Vertex vertices[] =
{
    sf::Vertex(sf::Vector2f( 0,    0), sf::Color::Red, sf::Vector2f( 0,    0)),
    sf::Vertex(sf::Vector2f( 0, 100), sf::Color::Red, sf::Vector2f( 0, 100)),
    sf::Vertex(sf::Vector2f(100, 100), sf::Color::Red, sf::Vector2f(10, 10)),
    sf::Vertex(sf::Vector2f(100,    0), sf::Color::Red, sf::Vector2f(10,    0)),
};

// draw it
window.draw(vertices, 4, sf::Quads);
```

Note: although texture coordinates are supposed to be an integer array, you can use floating-point values because of some buggy graphics drivers that are not able to process integers correctly.

See also

`sf::VertexArray`

Definition at line 42 of file `Vertex.hpp`.

Constructor & Destructor Documentation

sf::Vertex::Vertex ()

Default constructor.

sf::Vertex::Vertex (const Vector2f & thePosition)

Construct the vertex from its position.

The vertex color is white and texture coordinates are (0, 0).

Parameters

thePosition `Vertex` position

sf::Vertex::Vertex (const Vector2f & thePosition, const Color & theColor)

Construct the vertex from its position and color.

The texture coordinates are (0, 0).

Parameters

thePosition `Vertex` position
theColor `Vertex` color

```
sf::Vertex::Vertex ( const Vector2f & thePosition,  
                    const Vector2f & theTexCoords  
                )
```

Construct the vertex from its position and texture coordinates.

The vertex color is white.

Parameters

thePosition Vertex position
theTexCoords Vertex texture coordinates

```
sf::Vertex::Vertex ( const Vector2f & thePosition,  
                    const Color & theColor,  
                    const Vector2f & theTexCoords  
                )
```

Construct the vertex from its position, color and texture coordinates.

Parameters

thePosition Vertex position
theColor Vertex color
theTexCoords Vertex texture coordinates

Member Data Documentation

Color sf::Vertex::color

Color of the vertex.

Definition at line [98](#) of file [Vertex.hpp](#).

Vector2f sf::Vertex::position

2D position of the vertex

Definition at line [97](#) of file [Vertex.hpp](#).

Vector2f sf::Vertex::texCoords

Coordinates of the texture's pixel to map to the vertex.

Definition at line [99](#) of file [Vertex.hpp](#).

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

- [Vertex.hpp](#)
-

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [List of all members](#)

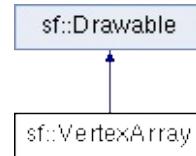
sf::VertexArray Class Reference

[Graphics module](#)

Define a set of one or more 2D primitives. More...

```
#include <VertexArray.hpp>
```

Inheritance diagram for sf::VertexArray:



Public Member Functions

`VertexArray ()`

Default constructor. [More...](#)

`VertexArray (PrimitiveType type, std::size_t vertexCount)`

Construct the vertex array with a type and an initial number of vertices.

`std::size_t getVertexCount () const`

Return the vertex count. [More...](#)

`Vertex & operator[] (std::size_t index)`

Get a read-write access to a vertex by its index. [More...](#)

`const Vertex & operator[] (std::size_t index) const`

Get a read-only access to a vertex by its index. [More...](#)

`void clear ()`

Clear the vertex array. [More...](#)

`void resize (std::size_t vertexCount)`

Resize the vertex array. [More...](#)

`void append (const Vertex &vertex)`

Add a vertex to the array. [More...](#)

`void setPrimitiveType (PrimitiveType type)`

Set the type of primitives to draw. [More...](#)

`PrimitiveType getPrimitiveType () const`

Get the type of primitives drawn by the vertex array. [More...](#)

`FloatRect getBounds () const`

Compute the bounding rectangle of the vertex array. More

Detailed Description

Define a set of one or more 2D primitives.

`sf::VertexArray` is a very simple wrapper around a dynamic array of vertices.

It inherits `sf::Drawable`, but unlike other drawables it is not transformable.

Example:

```
sf::VertexArray lines(sf::LineStrip, 4);
lines[0].position = sf::Vector2f(10, 0);
lines[1].position = sf::Vector2f(20, 0);
lines[2].position = sf::Vector2f(30, 5);
lines[3].position = sf::Vector2f(40, 2);

window.draw(lines);
```

See also

`sf::Vertex`

Definition at line 45 of file `VertexArray.hpp`.

Constructor & Destructor Documentation

`sf::VertexArray::VertexArray()`

Default constructor.

Creates an empty vertex array.

`sf::VertexArray::VertexArray(PrimitiveType type, std::size_t vertexCount = 0)`

Construct the vertex array with a type and an initial number of vertices.

Parameters

type Type of primitives

vertexCount Initial number of vertices in the array

Member Function Documentation

void sf::VertexArray::append (const Vertex & vertex)

Add a vertex to the array.

Parameters

vertex `Vertex` to add

void sf::VertexArray::clear ()

Clear the vertex array.

This function removes all the vertices from the array. It doesn't deallocate that adding new vertices after clearing doesn't involve reallocating all the

FloatRect sf::VertexArray::getBounds () const

Compute the bounding rectangle of the vertex array.

This function returns the minimal axis-aligned rectangle that contains all

Returns

Bounding rectangle of the vertex array

PrimitiveType `sf::VertexArray::getPrimitiveType() const`

Get the type of primitives drawn by the vertex array.

Returns

Primitive type

std::size_t `sf::VertexArray::getVertexCount() const`

Return the vertex count.

Returns

Number of vertices in the array

Vertex& `sf::VertexArray::operator[](std::size_t index)`

Get a read-write access to a vertex by its index.

This function doesn't check *index*, it must be in range [0, `getVertexCount()`]. It returns `sf::Vertex` undefined otherwise.

Parameters

index Index of the vertex to get

Returns

Reference to the index-th vertex

See also

`getVertexCount`

`const Vertex& sf::VertexArray::operator[] (std::size_t index) const`

Get a read-only access to a vertex by its index.

This function doesn't check *index*, it must be in range [0, `getVertexCount()`] undefined otherwise.

Parameters

index Index of the vertex to get

Returns

Const reference to the index-th vertex

See also

`getVertexCount()`

`void sf::VertexArray::resize (std::size_t vertexCount)`

Resize the vertex array.

If *vertexCount* is greater than the current size, the previous vertices (constructed) vertices are added. If *vertexCount* is less than the current size, vertices are removed from the array.

Parameters

vertexCount New size of the array (number of vertices)

`void sf::VertexArray::setPrimitiveType (PrimitiveType type)`

Set the type of primitives to draw.

This function defines how the vertices must be interpreted when it's time

- As points
- As lines
- As triangles
- As quads The default primitive type is sf::Points.

Parameters

type Type of primitive

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

- `VertexArray.hpp`

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [List of all members](#)

sf::View Class Reference

[Graphics module](#)

2D camera that defines what region is shown on screen [More...](#)

```
#include <View.hpp>
```

Public Member Functions

`View ()`

Default constructor. [More...](#)

`View (const FloatRect &rectangle)`

Construct the view from a rectangle. [More...](#)

`View (const Vector2f ¢er, const Vector2f &size)`

Construct the view from its center and size. [More...](#)

`void setCenter (float x, float y)`

Set the center of the view. [More...](#)

`void setCenter (const Vector2f ¢er)`

Set the center of the view. [More...](#)

`void setSize (float width, float height)`

Set the size of the view. [More...](#)

`void setSize (const Vector2f &size)`

Set the size of the view. [More...](#)

`void setRotation (float angle)`

Set the orientation of the view. [More...](#)

`void setViewport (const FloatRect &viewport)`

Set the target viewport. [More...](#)

`void reset (const FloatRect &rectangle)`

Reset the view to the given rectangle. [More...](#)

`const Vector2f & getCenter () const`

Get the center of the view. More...

const Vector2f & **getSize () const**
Get the size of the view. More...

float **getRotation () const**
Get the current orientation of the view. More...

const FloatRect & **getViewport () const**
Get the target viewport rectangle of the view. More...

void **move (float offsetX, float offsetY)**
Move the view relatively to its current position. More...

void **move (const Vector2f &offset)**
Move the view relatively to its current position. More...

void **rotate (float angle)**
Rotate the view relatively to its current orientation. M

void **zoom (float factor)**
Resize the view rectangle relatively to its current size

const Transform & **getTransform () const**
Get the projection transform of the view. More...

const Transform & **getInverseTransform () const**
Get the inverse projection transform of the view. Mor

Detailed Description

2D camera that defines what region is shown on screen

`sf::View` defines a camera in the 2D scene.

This is a very powerful concept: you can scroll, rotate or zoom the entire scene so your drawable objects are drawn.

A view is composed of a source rectangle, which defines what part of the source rectangle will be drawn, and a viewport, which defines where the contents of the source rectangle will be drawn (window or texture).

The viewport allows to map the scene to a custom part of the render target, such as the screen or for displaying a minimap, for example. If the source rectangle covers a larger area than the viewport, its contents will be stretched to fit in.

To apply a view, you have to assign it to the render target. Then, objects will be affected by the view until you use another view.

Usage example:

```
sf::RenderWindow window;
sf::View view;

// Initialize the view to a rectangle located at (100, 100) and with a
// width and height of 400 and 200 respectively
view.reset(sf::FloatRect(100, 100, 400, 200));

// Rotate it by 45 degrees
view.rotate(45);

// Set its target viewport to be half of the window
view.setViewport(sf::FloatRect(0.f, 0.f, 0.5f, 1.f));

// Apply it
window.setView(view);

// Render stuff
```

```
window.draw(someSprite);

// Set the default view back
window.setView(window.getDefaultView());

// Render stuff not affected by the view
window.draw(someText);
```

See also the note on coordinates and undistorted rendering in [sf::Transform](#).

See also

[sf::RenderWindow](#), [sf::RenderTexture](#)

Definition at line 43 of file [View.hpp](#).

Constructor & Destructor Documentation

`sf::View::View()`

Default constructor.

This constructor creates a default view of (0, 0, 1000, 1000)

`sf::View::View(const FloatRect & rectangle)`

Construct the view from a rectangle.

Parameters

rectangle Rectangle defining the zone to display

`sf::View::View(const Vector2f & center, const Vector2f & size)`

Construct the view from its center and size.

Parameters

center Center of the zone to display
size Size of zone to display

Member Function Documentation

const Vector2f& sf::View::getCenter() const

Get the center of the view.

Returns

Center of the view

See also

[getSize](#), [setCenter](#)

const Transform& sf::View::getInverseTransform() const

Get the inverse projection transform of the view.

This function is meant for internal use only.

Returns

Inverse of the projection transform defining the view

See also

[getTransform](#)

float sf::View::getRotation() const

Get the current orientation of the view.

Returns

Rotation angle of the view, in degrees

See also

[setRotation](#)

const `Vector2f&` sf::View::getSize () const

Get the size of the view.

Returns

Size of the view

See also

[getCenter](#), [setSize](#)

const `Transform&` sf::View::getTransform () const

Get the projection transform of the view.

This function is meant for internal use only.

Returns

Projection transform defining the view

See also

[getInverseTransform](#)

const **FloatRect&** **sf::View::getViewport() const**

Get the target viewport rectangle of the view.

Returns

Viewport rectangle, expressed as a factor of the target size

See also

[setViewport](#)

void sf::View::move (float offsetX, float offsetY)

Move the view relatively to its current position.

Parameters

offsetX X coordinate of the move offset

offsetY Y coordinate of the move offset

See also

[setCenter](#), [rotate](#), [zoom](#)

void sf::View::move (const Vector2f & offset)

Move the view relatively to its current position.

Parameters

offset Move offset

See also

[setCenter](#), [rotate](#), [zoom](#)

void sf::View::reset(const FloatRect & rectangle)

Reset the view to the given rectangle.

Note that this function resets the rotation angle to 0.

Parameters

rectangle Rectangle defining the zone to display

See also

[setCenter](#), [setSize](#), [setRotation](#)

void sf::View::rotate(float angle)

Rotate the view relatively to its current orientation.

Parameters

angle Angle to rotate, in degrees

See also

[setRotation](#), [move](#), [zoom](#)

void sf::View::setCenter(float x, float y)

Set the center of the view.

Parameters

x X coordinate of the new center

y Y coordinate of the new center

See also

[setSize](#), [getCenter](#)

void sf::View::setCenter (const Vector2f & center)

Set the center of the view.

Parameters

center New center

See also

[setSize](#), [getCenter](#)

void sf::View::setRotation (float angle)

Set the orientation of the view.

The default rotation of a view is 0 degree.

Parameters

angle New angle, in degrees

See also

[getRotation](#)

```
void sf::View::setSize ( float width,  
                        float height  
)
```

Set the size of the view.

Parameters

width New width of the view
height New height of the view

See also

[setCenter](#), [getCenter](#)

```
void sf::View::setSize ( const Vector2f & size )
```

Set the size of the view.

Parameters

size New size

See also

[setCenter](#), [getCenter](#)

```
void sf::View::setViewport ( const FloatRect & viewport )
```

Set the target viewport.

The viewport is the rectangle into which the contents of the view are drawn (between 0 and 1) of the size of the [RenderTarget](#) to which the view is attached.

which takes the left side of the target would be defined with `View.setViewport`. By default, a view has a viewport which covers the entire target.

Parameters

viewport New viewport rectangle

See also

`getViewport`

void sf::View::zoom (float factor)

Resize the view rectangle relatively to its current size.

Resizing the view simulates a zoom, as the zone displayed on screen is multiplied by a zoom factor. The zoom factor is a floating-point multiplier:

- 1 keeps the size unchanged
- > 1 makes the view bigger (objects appear smaller)
- < 1 makes the view smaller (objects appear bigger)

Parameters

factor Zoom factor to apply

See also

`setSize`, `move`, `rotate`

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

- `View.hpp`

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

Classes

Network module

Socket-based communication, utilities and higher-level network protocols

Classes

class [sf::Ftp](#)
A FTP client. [More...](#)

class [sf::Http](#)
A HTTP client. [More...](#)

class [sf::IpAddress](#)
Encapsulate an IPv4 network address. [More...](#)

class [sf::Packet](#)
Utility class to build blocks of data to transfer over the network. [M](#)

class [sf::Socket](#)
Base class for all the socket types. [More...](#)

class [sf::SocketSelector](#)
Multiplexer that allows to read from multiple sockets. [More...](#)

class [sf::TcpListener](#)
Socket that listens to new TCP connections. [More...](#)

class [sf::TcpSocket](#)
Specialized socket using the TCP protocol. [More...](#)

class [sf::UdpSocket](#)
Specialized socket using the UDP protocol. [More...](#)

Detailed Description

Socket-based communication, utilities and higher-level network protocols

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Classes](#) | [Public Types](#) | [Public Member Functions](#) | [Friends](#) | [List of all members](#)

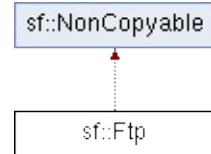
sf::Ftp Class Reference

Network module

A FTP client. [More...](#)

```
#include <Ftp.hpp>
```

Inheritance diagram for sf::Ftp:



Classes

class [DirectoryResponse](#)

Specialization of FTP response returning a directory. [More...](#)

class [ListingResponse](#)

Specialization of FTP response returning a filename listing. [More...](#)

class [Response](#)

Define a FTP response. [More...](#)

Public Types

enum TransferMode { Binary, Ascii, Ebcdic }
Enumeration of transfer modes. [More...](#)

Public Member Functions

`~Ftp ()`

Destructor. [More...](#)

Response `connect (const IpAddress &server, unsigned short port, const Time::TimeSpan &timeout=Time::Zero)`
Connect to the specified FTP server. [More...](#)

Response `disconnect ()`

Close the connection with the server. [More...](#)

Response `login ()`

Log in using an anonymous account. [More...](#)

Response `login (const std::string &name, const std::string &password)`
Log in using a username and a password. [More...](#)

Response `keepAlive ()`

Send a null command to keep the connection alive.

DirectoryResponse `getWorkingDirectory ()`

Get the current working directory. [More...](#)

ListingResponse `getDirectoryListing (const std::string &directory="")`
Get the contents of the given directory. [More...](#)

Response `changeDirectory (const std::string &directory)`

Change the current working directory. [More...](#)

Response `parentDirectory ()`

Go to the parent directory of the current one. [More...](#)

Response `createDirectory` (const std::string &name)
Create a new directory. [More...](#)

Response `deleteDirectory` (const std::string &name)
Remove an existing directory. [More...](#)

Response `renameFile` (const std::string &file, const std::string &newName)
Rename an existing file. [More...](#)

Response `deleteFile` (const std::string &name)
Remove an existing file. [More...](#)

Response `download` (const std::string &remoteFile, const std::string &localFile, mode=Binary)
Download a file from the server. [More...](#)

Response `upload` (const std::string &localFile, const std::string &remoteFile, mode=Binary)
Upload a file to the server. [More...](#)

Response `sendCommand` (const std::string &command, const std::string ¶meters)
Send a command to the FTP server. [More...](#)

Friends

class **DataChannel**

Detailed Description

A FTP client.

`sf::Ftp` is a very simple FTP client that allows you to communicate with a F

The FTP protocol allows you to manipulate a remote file system (list remove, ...).

Using the FTP client consists of 4 parts:

- Connecting to the FTP server
- Logging in (either as a registered user or anonymously)
- Sending commands to the server
- Disconnecting (this part can be done implicitly by the destructor)

Every command returns a FTP response, which contains the status code of the server. Some commands such as `getWorkingDirectory()` and `getDirectc` and use a class derived from `sf::Ftp::Response` to provide this data. The `sendCommand()` function is directly provided as member functions, but it is also possible to use the `sendCommand()` function.

Note that response statuses ≥ 1000 are not part of the FTP standard, they indicate an internal error occurs.

All commands, especially upload and download, may take some time to complete. If you know if you don't want to block your application while the server is comple

Usage example:

```
// Create a new FTP client
```

```

sf::Ftp ftp;

// Connect to the server
sf::Ftp::Response response = ftp.connect("ftp://ftp.myserver.com");
if (response.isOk())
    std::cout << "Connected" << std::endl;

// Log in
response = ftp.login("laurent", "dF6Zm89D");
if (response.isOk())
    std::cout << "Logged in" << std::endl;

// Print the working directory
sf::Ftp::DirectoryResponse directory = ftp.getWorkingDirectory();
if (directory.isOk())
    std::cout << "Working directory: " << directory.getDirectory() << std::endl;

// Create a new directory
response = ftp.createDirectory("files");
if (response.isOk())
    std::cout << "Created new directory" << std::endl;

// Upload a file to this new directory
response = ftp.upload("local-path/file.txt", "files", sf::Ftp::Ascii);
if (response.isOk())
    std::cout << "File uploaded" << std::endl;

// Send specific commands (here: FEAT to list supported FTP features)
response = ftp.sendCommand("FEAT");
if (response.isOk())
    std::cout << "Feature list:\n" << response.getMessage() << std::endl;

// Disconnect from the server (optional)
ftp.disconnect();

```

Definition at line 47 of file Ftp.hpp.

Member Enumeration Documentation

enum sf::Ftp::TransferMode

Enumeration of transfer modes.

Enumerator	
Binary	Binary mode (file is transferred as a sequence of bytes)
Ascii	Text mode using ASCII encoding.
Ebcdic	Text mode using EBCDIC encoding.

Definition at line 55 of file [Ftp.hpp](#).

Constructor & Destructor Documentation

sf::Ftp::~Ftp()

Destructor.

Automatically closes the connection with the server if it is still opened.

Member Function Documentation

Response `sf::Ftp::changeDirectory (const std::string & directory)`

Change the current working directory.

The new directory must be relative to the current one.

Parameters

`directory` New working directory

Returns

Server response to the request

See also

`getWorkingDirectory`, `getDirectoryListing`, `parentDirectory`

Response `sf::Ftp::connect (const IpAddress & server, unsigned short port = 21, Time timeout = Time::Zero)`

Connect to the specified FTP server.

The port has a default value of 21, which is the standard port used by. If you want to connect to a different port, you can use a different value, unless you really know what you do. This function may take a while to complete, especially if the server is not ready. If your application needs to wait for too long, you can use a timeout. The default value, `Time::Zero`, means that there is no timeout.

timeout will be used (which is usually pretty long).

Parameters

server Name or address of the FTP server to connect to

port Port used for the connection

timeout Maximum time to wait

Returns

Server response to the request

See also

[disconnect](#)

Response `sf::Ftp::createDirectory (const std::string & name)`

Create a new directory.

The new directory is created as a child of the current working directory.

Parameters

name Name of the directory to create

Returns

Server response to the request

See also

[deleteDirectory](#)

Response `sf::Ftp::deleteDirectory (const std::string & name)`

Remove an existing directory.

The directory to remove must be relative to the current working directory.
the directory will be removed permanently!

Parameters

name Name of the directory to remove

Returns

Server response to the request

See also

[createDirectory](#)

Response `sf::Ftp::deleteFile (const std::string & name)`

Remove an existing file.

The file name must be relative to the current working directory. Use this
be removed permanently!

Parameters

name File to remove

Returns

Server response to the request

See also

[renameFile](#)

Response `sf::Ftp::disconnect ()`

Close the connection with the server.

Returns

Server response to the request

See also

[connect](#)

```
Response sf::Ftp::download ( const std::string & remoteFile,
                             const std::string & localPath,
                             TransferMode      mode = Binary
                           )
```

Download a file from the server.

The filename of the distant file is relative to the current working directory. The destination path is relative to the current directory of your application. If the distant file already exists in the local destination path, it will be overwritten.

Parameters

remoteFile Filename of the distant file to download

localPath The directory in which to put the file on the local computer

mode Transfer mode

Returns

Server response to the request

See also

[upload](#)

```
ListingResponse sf::Ftp::getDirectoryListing ( const std::string & d
```

Get the contents of the given directory.

This function retrieves the sub-directories and files contained in the given directory.
The *directory* parameter is relative to the current working directory.

Parameters

directory Directory to list

Returns

Server response to the request

See also

[getWorkingDirectory](#), [changeDirectory](#), [parentDirectory](#)

DirectoryResponse **sf::Ftp::getWorkingDirectory ()**

Get the current working directory.

The working directory is the root path for subsequent operations involving the file system.

Returns

Server response to the request

See also

[getDirectoryListing](#), [changeDirectory](#), [parentDirectory](#)

Response **sf::Ftp::keepAlive ()**

Send a null command to keep the connection alive.

This command is useful because the server may close the connection sent.

Returns

Server response to the request

Response sf::Ftp::login ()

Log in using an anonymous account.

Logging in is mandatory after connecting to the server. Users that are n operation.

Returns

Server response to the request

Response sf::Ftp::login (const std::string & name, const std::string & password)

Log in using a username and a password.

Logging in is mandatory after connecting to the server. Users that are n operation.

Parameters

name User name
password Password

Returns

Server response to the request

Response sf::Ftp::parentDirectory ()

Go to the parent directory of the current one.

Returns

Server response to the request

See also

[getWorkingDirectory](#), [getDirectoryListing](#), [changeDirectory](#)

Response sf::Ftp::renameFile (const std::string & file, const std::string & newName)

Rename an existing file.

The filenames must be relative to the current working directory.

Parameters

file File to rename

newName New name of the file

Returns

Server response to the request

See also

[deleteFile](#)

Response sf::Ftp::sendCommand (const std::string & command,

```
const std::string & parameter =  
)
```

Send a command to the FTP server.

While the most often used commands are provided as member functions can be used to send any FTP command to the server. If the command re they can be specified in *parameter*. If the server returns information, you using `Response::getMessage()`.

Parameters

command Command to send
parameter Command parameter

Returns

Server response to the request

```
Response sf::Ftp::upload ( const std::string & localFile,  
                           const std::string & remotePath,  
                           TransferMode mode = Binary  
)
```

Upload a file to the server.

The name of the local file is relative to the current working directory of the process. The path is relative to the current directory of the FTP server.

Parameters

localFile Path of the local file to upload
remotePath The directory in which to put the file on the server
mode Transfer mode

Returns

Server response to the request

See also

[download](#)

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

- [Ftp.hpp](#)

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

Classes

[Files](#)

[Class List](#)

[Class Index](#)

[Class Hierarchy](#)

[Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [List of all members](#)

sf::Ftp::DirectoryResponse Class Reference

Specialization of FTP response returning a directory. [More...](#)

#include <Ftp.hpp>

Inheritance diagram for sf::Ftp::DirectoryResponse:



Public Types

```
enum Status {
    RestartMarkerReply = 110, ServiceReadySoon = 120, DataCor
    OpeningDataConnection = 150,
    Ok = 200, PointlessCommand = 202, SystemStatus = 211, Dire
    FileStatus = 213, HelpMessage = 214, SystemType = 215, Ser
    ClosingConnection = 221, DataConnectionOpened = 225, Clos
    EnteringPassiveMode = 227,
    LoggedIn = 230, FileActionOk = 250, DirectoryOk = 257, Needl
    NeedAccountToLogIn = 332, NeedInformation = 350, ServiceU
    DataConnectionUnavailable = 425,
    TransferAborted = 426, FileActionAborted = 450, LocalError = 4
    452,
    CommandUnknown = 500, ParametersUnknown = 501, Comm
    BadCommandSequence = 503,
    ParameterNotImplemented = 504, NotLoggedIn = 530, NeedAc
    FileUnavailable = 550,
    PageTypeUnknown = 551, NotEnoughMemory = 552, Filenam
    InvalidResponse = 1000,
    ConnectionFailed = 1001, ConnectionClosed = 1002, InvalidFil
}
```

Status codes possibly returned by a FTP response. [More...](#)

Public Member Functions

`DirectoryResponse (const Response &response)`
Default constructor. [More...](#)

`const std::string & getDirectory () const`
Get the directory returned in the response. [More...](#)

`bool isOk () const`
Check if the status code means a success. [More...](#)

`Status getStatus () const`
Get the status code of the response. [More...](#)

`const std::string & getMessage () const`
Get the full message contained in the response. [More...](#)

Detailed Description

Specialization of FTP response returning a directory.

Definition at line [188](#) of file [Ftp.hpp](#).

Member Enumeration Documentation

enum sf::Ftp::Response::Status

Status codes possibly returned by a FTP response.

Enumerator	
RestartMarkerReply	Restart marker reply.
ServiceReadySoon	Service ready in N minutes.
DataConnectionAlreadyOpened	Data connection already opened, tra
OpeningDataConnection	File status ok, about to open data cor
Ok	Command ok.
PointlessCommand	Command not implemented.
SystemStatus	System status, or system help reply.
DirectoryStatus	Directory status.
FileStatus	

	File status.
HelpMessage	Help message.
SystemType	NAME system type, where NAME is the list in the Assigned Numbers doc
ServiceReady	Service ready for new user.
ClosingConnection	Service closing control connection.
DataConnectionOpened	Data connection open, no transfer in
ClosingDataConnection	Closing data connection, requested fi
EnteringPassiveMode	Entering passive mode.
LoggedIn	User logged in, proceed. Logged out
FileActionOk	Requested file action ok.
DirectoryOk	PATHNAME created.
NeedPassword	User name ok, need password.
NeedAccountToLogIn	Need account for login.

NeedInformation	Requested file action pending further
ServiceUnavailable	Service not available, closing control
DataConnectionUnavailable	Can't open data connection.
TransferAborted	Connection closed, transfer aborted.
FileActionAborted	Requested file action not taken.
LocalError	Requested action aborted, local error
InsufficientStorageSpace	Requested action not taken; insufficient storage space available.
CommandUnknown	Syntax error, command unrecognized
ParametersUnknown	Syntax error in parameters or arguments
CommandNotImplemented	Command not implemented.
BadCommandSequence	Bad sequence of commands.
ParameterNotImplemented	Command not implemented for that parameter
NotLoggedIn	Not logged in.

NeedAccountToStore	Need account for storing files.
FileUnavailable	Requested action not taken, file unav
PageTypeUnknown	Requested action aborted, page type
NotEnoughMemory	Requested file action aborted, exceede
FilenameNotAllowed	Requested action not taken, file name
InvalidResponse	Not part of the FTP standard, genera response cannot be parsed.
ConnectionFailed	Not part of the FTP standard, gene level socket connection with the serv
ConnectionClosed	Not part of the FTP standard, gene level socket connection is unexpected
InvalidFile	Not part of the FTP standard, genera cannot be read or written.

Definition at line 74 of file [Ftp.hpp](#).

Constructor & Destructor Documentation

sf::Ftp::DirectoryResponse::DirectoryResponse (const Response &

Default constructor.

Parameters

response Source response

Member Function Documentation

const std::string& sf::Ftp::DirectoryResponse::getDirectory() const

Get the directory returned in the response.

Returns

Directory name

const std::string& sf::Ftp::Response::getMessage() const

Get the full message contained in the response.

Returns

The response message

Status sf::Ftp::Response::getStatus() const

Get the status code of the response.

Returns

Status code

bool sf::Ftp::Response::isOk() const

Check if the status code means a success.

This function is defined for convenience, it is equivalent to testing if the s

Returns

True if the status is a success, false if it is a failure

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

- [Ftp.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

Classes

[Files](#)

[Class List](#)

[Class Index](#)

[Class Hierarchy](#)

[Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [List of all members](#)

sf::Ftp::ListingResponse Class Reference

Specialization of FTP response returning a filename listing. [More...](#)

```
#include <Ftp.hpp>
```

Inheritance diagram for sf::Ftp::ListingResponse:



Public Types

```
enum Status {
    RestartMarkerReply = 110, ServiceReadySoon = 120, DataCor
    OpeningDataConnection = 150,
    Ok = 200, PointlessCommand = 202, SystemStatus = 211, Dire
    FileStatus = 213, HelpMessage = 214, SystemType = 215, Ser
    ClosingConnection = 221, DataConnectionOpened = 225, Clos
    EnteringPassiveMode = 227,
    LoggedIn = 230, FileActionOk = 250, DirectoryOk = 257, Needl
    NeedAccountToLogIn = 332, NeedInformation = 350, ServiceU
    DataConnectionUnavailable = 425,
    TransferAborted = 426, FileActionAborted = 450, LocalError = 4
    452,
    CommandUnknown = 500, ParametersUnknown = 501, Comm
    BadCommandSequence = 503,
    ParameterNotImplemented = 504, NotLoggedIn = 530, NeedAc
    FileUnavailable = 550,
    PageTypeUnknown = 551, NotEnoughMemory = 552, Filenam
    InvalidResponse = 1000,
    ConnectionFailed = 1001, ConnectionClosed = 1002, InvalidFil
}
```

Status codes possibly returned by a FTP response. [More...](#)

Public Member Functions

`ListingResponse (const Response &re`
Default constructor. More...

`const std::vector< std::string > & getListing () const`
Return the array of directory/file names

`bool isOk () const`
Check if the status code means a succ

`Status getStatus () const`
Get the status code of the response. M

`const std::string & getMessage () const`
Get the full message contained in the r

Detailed Description

Specialization of FTP response returning a filename listing.

Definition at line [221](#) of file [Ftp.hpp](#).

Member Enumeration Documentation

enum sf::Ftp::Response::Status

Status codes possibly returned by a FTP response.

Enumerator	
RestartMarkerReply	Restart marker reply.
ServiceReadySoon	Service ready in N minutes.
DataConnectionAlreadyOpened	Data connection already opened, tra
OpeningDataConnection	File status ok, about to open data cor
Ok	Command ok.
PointlessCommand	Command not implemented.
SystemStatus	System status, or system help reply.
DirectoryStatus	Directory status.
FileStatus	

	File status.
HelpMessage	Help message.
SystemType	NAME system type, where NAME is the list in the Assigned Numbers doc
ServiceReady	Service ready for new user.
ClosingConnection	Service closing control connection.
DataConnectionOpened	Data connection open, no transfer in
ClosingDataConnection	Closing data connection, requested fi
EnteringPassiveMode	Entering passive mode.
LoggedIn	User logged in, proceed. Logged out
FileActionOk	Requested file action ok.
DirectoryOk	PATHNAME created.
NeedPassword	User name ok, need password.
NeedAccountToLogIn	Need account for login.

NeedInformation	Requested file action pending further
ServiceUnavailable	Service not available, closing control
DataConnectionUnavailable	Can't open data connection.
TransferAborted	Connection closed, transfer aborted.
FileActionAborted	Requested file action not taken.
LocalError	Requested action aborted, local error
InsufficientStorageSpace	Requested action not taken; insufficient storage space available.
CommandUnknown	Syntax error, command unrecognized
ParametersUnknown	Syntax error in parameters or arguments
CommandNotImplemented	Command not implemented.
BadCommandSequence	Bad sequence of commands.
ParameterNotImplemented	Command not implemented for that parameter
NotLoggedIn	Not logged in.

NeedAccountToStore	Need account for storing files.
FileUnavailable	Requested action not taken, file unav
PageTypeUnknown	Requested action aborted, page type
NotEnoughMemory	Requested file action aborted, exceede
FilenameNotAllowed	Requested action not taken, file name
InvalidResponse	Not part of the FTP standard, genera response cannot be parsed.
ConnectionFailed	Not part of the FTP standard, gene level socket connection with the serv
ConnectionClosed	Not part of the FTP standard, gene level socket connection is unexpected
InvalidFile	Not part of the FTP standard, genera cannot be read or written.

Definition at line 74 of file [Ftp.hpp](#).

Constructor & Destructor Documentation

```
sf::Ftp::ListingResponse::ListingResponse ( const Response & res
                                           const std::string & dat
                                         )
```

Default constructor.

Parameters

response	Source response
data	Data containing the raw listing

Member Function Documentation

const std::vector<std::string>& sf::Ftp::ListingResponse::getListing()

Return the array of directory/file names.

Returns

Array containing the requested listing

const std::string& sf::Ftp::Response::getMessage() const

Get the full message contained in the response.

Returns

The response message

Status sf::Ftp::Response::getStatus() const

Get the status code of the response.

Returns

Status code

bool sf::Ftp::Response::isOk() const

Check if the status code means a success.

This function is defined for convenience, it is equivalent to testing if the s

Returns

True if the status is a success, false if it is a failure

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

- [Ftp.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

Classes

[Files](#)

[Class List](#)

[Class Index](#)

[Class Hierarchy](#)

[Class Members](#)

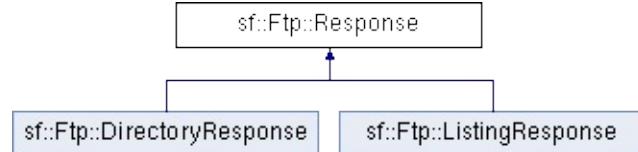
[Public Types](#) | [Public Member Functions](#) | [List of all members](#)

sf::Ftp::Response Class Reference

Define a FTP response. [More...](#)

```
#include <Ftp.hpp>
```

Inheritance diagram for sf::Ftp::Response:



Public Types

```
enum Status {
    RestartMarkerReply = 110, ServiceReadySoon = 120, DataCor
    OpeningDataConnection = 150,
    Ok = 200, PointlessCommand = 202, SystemStatus = 211, Dire
    FileStatus = 213, HelpMessage = 214, SystemType = 215, Ser
    ClosingConnection = 221, DataConnectionOpened = 225, Clos
    EnteringPassiveMode = 227,
    LoggedIn = 230, FileActionOk = 250, DirectoryOk = 257, Needl
    NeedAccountToLogIn = 332, NeedInformation = 350, ServiceU
    DataConnectionUnavailable = 425,
    TransferAborted = 426, FileActionAborted = 450, LocalError = 4
    452,
    CommandUnknown = 500, ParametersUnknown = 501, Comm
    BadCommandSequence = 503,
    ParameterNotImplemented = 504, NotLoggedIn = 530, NeedAc
    FileUnavailable = 550,
    PageTypeUnknown = 551, NotEnoughMemory = 552, Filenam
    InvalidResponse = 1000,
    ConnectionFailed = 1001, ConnectionClosed = 1002, InvalidFil
}
```

Status codes possibly returned by a FTP response. [More...](#)

Public Member Functions

Response (`Status code=InvalidResponse, const std::string & message`)
Default constructor. [More...](#)

bool `isOk () const`
Check if the status code means a success. [More...](#)

Status `getStatus () const`
Get the status code of the response. [More...](#)

const std::string & `getMessage () const`
Get the full message contained in the response. [More...](#)

Detailed Description

Define a FTP response.

Definition at line [66](#) of file [Ftp.hpp](#).

Member Enumeration Documentation

enum sf::Ftp::Response::Status

Status codes possibly returned by a FTP response.

Enumerator	
RestartMarkerReply	Restart marker reply.
ServiceReadySoon	Service ready in N minutes.
DataConnectionAlreadyOpened	Data connection already opened, tra
OpeningDataConnection	File status ok, about to open data cor
Ok	Command ok.
PointlessCommand	Command not implemented.
SystemStatus	System status, or system help reply.
DirectoryStatus	Directory status.
FileStatus	

	File status.
HelpMessage	Help message.
SystemType	NAME system type, where NAME is the list in the Assigned Numbers doc
ServiceReady	Service ready for new user.
ClosingConnection	Service closing control connection.
DataConnectionOpened	Data connection open, no transfer in
ClosingDataConnection	Closing data connection, requested fi
EnteringPassiveMode	Entering passive mode.
LoggedIn	User logged in, proceed. Logged out
FileActionOk	Requested file action ok.
DirectoryOk	PATHNAME created.
NeedPassword	User name ok, need password.
NeedAccountToLogIn	Need account for login.

NeedInformation	Requested file action pending further
ServiceUnavailable	Service not available, closing control
DataConnectionUnavailable	Can't open data connection.
TransferAborted	Connection closed, transfer aborted.
FileActionAborted	Requested file action not taken.
LocalError	Requested action aborted, local error
InsufficientStorageSpace	Requested action not taken; insufficient storage space available.
CommandUnknown	Syntax error, command unrecognized
ParametersUnknown	Syntax error in parameters or arguments
CommandNotImplemented	Command not implemented.
BadCommandSequence	Bad sequence of commands.
ParameterNotImplemented	Command not implemented for that parameter
NotLoggedIn	Not logged in.

NeedAccountToStore	Need account for storing files.
FileUnavailable	Requested action not taken, file unav
PageTypeUnknown	Requested action aborted, page type
NotEnoughMemory	Requested file action aborted, exceede
FilenameNotAllowed	Requested action not taken, file name
InvalidResponse	Not part of the FTP standard, genera response cannot be parsed.
ConnectionFailed	Not part of the FTP standard, gene level socket connection with the serv
ConnectionClosed	Not part of the FTP standard, gene level socket connection is unexpected
InvalidFile	Not part of the FTP standard, genera cannot be read or written.

Definition at line 74 of file [Ftp.hpp](#).

Constructor & Destructor Documentation

```
sf::Ftp::Response::Response ( Status code = InvalidRes
                             const std::string & message = ""
                             )
```

Default constructor.

This constructor is used by the FTP client to build the response.

Parameters

code Response status code
message Response message

Member Function Documentation

const std::string& sf::Ftp::Response::getMessage() const

Get the full message contained in the response.

Returns

The response message

Status sf::Ftp::Response::getStatus() const

Get the status code of the response.

Returns

Status code

bool sf::Ftp::Response::isOk() const

Check if the status code means a success.

This function is defined for convenience, it is equivalent to testing if the s

Returns

True if the status is a success, false if it is a failure

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

- [Ftp.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

Classes

[Files](#)

[Class List](#)

[Class Index](#)

[Class Hierarchy](#)

[Class Members](#)

[Classes](#) | [Public Member Functions](#) | [List of all members](#)

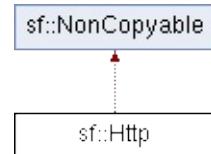
sf::Http Class Reference

[Network module](#)

A HTTP client. More...

```
#include <Http.hpp>
```

Inheritance diagram for sf::Http:



Classes

class **Request**

Define a HTTP request. [More...](#)

class **Response**

Define a HTTP response. [More...](#)

Public Member Functions

`Http ()`

Default constructor. [More...](#)

`Http (const std::string &host, unsigned short port=0)`

Construct the HTTP client with the target host. [More...](#)

`void setHost (const std::string &host, unsigned short port=0)`

Set the target host. [More...](#)

`Response sendRequest (const Request &request, Time timeout=Time::`

Send a HTTP request and return the server's response. [More...](#)

Detailed Description

A HTTP client.

`sf::Http` is a very simple HTTP client that allows you to communicate with

You can retrieve web pages, send data to an interactive resource, download files, etc. The protocol is not supported.

The HTTP client is split into 3 classes:

- `sf::Http::Request`
- `sf::Http::Response`
- `sf::Http`

`sf::Http::Request` builds the request that will be sent to the server. A request contains:

- a method (what you want to do)
- a target URI (usually the name of the web page or file)
- one or more header fields (options that you can pass to the server)
- an optional body (for POST requests)

`sf::Http::Response` parse the response from the web server and provides information about it. A response contains:

- a status code
- header fields (that may be answers to the ones that you requested)
- a body, which contains the contents of the requested resource

`sf::Http` provides a simple function, `SendRequest`, to send a `sf::Http::Request` and receive a `sf::Http::Response`.

`sf::Http::Response` from the server.

Usage example:

```
// Create a new HTTP client
sf::Http http;

// We'll work on http://www.sfml-dev.org
http.setHost("http://www.sfml-dev.org");

// Prepare a request to get the 'features.php' page
sf::Http::Request request("features.php");

// Send the request
sf::Http::Response response = http.sendRequest(request);

// Check the status code and display the result
sf::Http::Response::Status status = response.getStatus();
if (status == sf::Http::Response::Ok)
{
    std::cout << response.getBody() << std::endl;
}
else
{
    std::cout << "Error " << status << std::endl;
}
```

Definition at line 46 of file `Http.hpp`.

Constructor & Destructor Documentation

`sf::Http::Http()`

Default constructor.

```
sf::Http::Http ( const std::string & host,  
                unsigned short      port = 0  
              )
```

Construct the HTTP client with the target host.

This is equivalent to calling `setHost(host, port)`. The port has a default value of 80. The HTTP client will use the right port according to the protocol used (80 for HTTP and 443 for HTTPS) unless you really need a port other than the standard one, or use an explicit port number.

Parameters

host Web server to connect to
port Port to use for connection

Member Function Documentation

```
Response sf::Http::sendRequest ( const Request & request,
                                Time               timeout = Time::Zero
                                )
```

Send a HTTP request and return the server's response.

You must have a valid host before sending a request (see `setHost`). Any `host` in the `request` will be added with an appropriate value. Warning: this function may block and may not return instantly; use a thread if you don't want to wait. The `timeout` parameter specifies the maximum time to wait. A value of `Time::Zero` means that the function will wait forever (which is usually pretty long).

Parameters

`request` Request to send

`timeout` Maximum time to wait

Returns

Server's response

```
void sf::Http::setHost ( const std::string & host,
                        unsigned short      port = 0
                        )
```

Set the target host.

This function just stores the host address and port, it doesn't actually

request. The port has a default value of 0, which means that the HT according to the protocol used (80 for HTTP). You should leave it like t other than the standard one, or use an unknown protocol.

Parameters

host Web server to connect to

port Port to use for connection

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

- [Http.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [Friends](#) | [List of all members](#)

sf::Http::Request Class Reference

Define a HTTP request. [More...](#)

```
#include <Http.hpp>
```

Public Types

```
Method {  
enum   Get, Post, Head, Put,  
        Delete  
}
```

Enumerate the available HTTP methods for a request. [More...](#)

Public Member Functions

`Request (const std::string &uri="/", Method method=Get, const std::string &body="")`
Default constructor. [More...](#)

`void setField (const std::string &field, const std::string &value)`
Set the value of a field. [More...](#)

`void setMethod (Method method)`
Set the request method. [More...](#)

`void setUri (const std::string &uri)`
Set the requested URI. [More...](#)

`void setHttpVersion (unsigned int major, unsigned int minor)`
Set the HTTP version for the request. [More...](#)

`void setBody (const std::string &body)`
Set the body of the request. [More...](#)

Friends

class **Http**

Detailed Description

Define a HTTP request.

Definition at line 54 of file [Http.hpp](#).

Member Enumeration Documentation

enum sf::Http::Request::Method

Enumerate the available HTTP methods for a request.

Enumerator	
Get	Request in get mode, standard method to retrieve a page.
Post	Request in post mode, usually to send data to a page.
Head	Request a page's header only.
Put	Request in put mode, useful for a REST API.
Delete	Request in delete mode, useful for a REST API.

Definition at line 62 of file [Http.hpp](#).

Constructor & Destructor Documentation

```
sf::Http::Request::Request ( const std::string & uri = "/",
                            Method               method = Get,
                            const std::string & body = ""
                           )
```

Default constructor.

This constructor creates a GET request, with the root URI ("") and an en

Parameters

uri Target URI

method Method to use for the request

body Content of the request's body

Member Function Documentation

```
void sf::Http::Request::setBody ( const std::string & body )
```

Set the body of the request.

The body of a request is optional and only makes sense for POST requests. The body is empty by default.

Parameters

body Content of the body

```
void sf::Http::Request::setField ( const std::string & field,
                                  const std::string & value
                                )
```

Set the value of a field.

The field is created if it doesn't exist. The name of the field is case-insensitive. If the request doesn't contain any field (but the mandatory fields are added later by the server), the field is added.

Parameters

field Name of the field to set

value Value of the field

```
void sf::Http::Request::setHttpVersion ( unsigned int major,  
                                         unsigned int minor  
                                         )
```

Set the HTTP version for the request.

The HTTP version is 1.0 by default.

Parameters

major Major HTTP version number
minor Minor HTTP version number

```
void sf::Http::Request::setMethod ( Method method )
```

Set the request method.

See the [Method](#) enumeration for a complete list of all the available methods. The [Http::Request::Get](#) method is used by default.

Parameters

method Method to use for the request

```
void sf::Http::Request::setUri ( const std::string & uri )
```

Set the requested URI.

The URI is the resource (usually a web page or a file) that you want to request. The [Http::Request::Get](#) method is used by default.

Parameters

uri URI to request, relative to the host

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

- [Http.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [Friends](#) | [List of all members](#)

sf::Http::Response Class Reference

Define a HTTP response. [More...](#)

```
#include <Http.hpp>
```

Public Types

```
Status {  
    Ok = 200, Created = 201, Accepted = 202, NoContent = 204,  
    ResetContent = 205, PartialContent = 206, MultipleChoices = 3  
    MovedTemporarily = 302, NotModified = 304, BadRequest = 40  
enum    Forbidden = 403, NotFound = 404, RangeNotSatisfiable = 407,  
        NotImplemented = 501, BadGateway = 502, ServiceUnavailable =  
        504,  
        VersionNotSupported = 505, InvalidResponse = 1000, Connect =  
        1200  
}  
Enumerate all the valid status codes for a response. More...
```

Enumerate all the valid status codes for a response. More...

Public Member Functions

`Response ()`

Default constructor. [More...](#)

`const std::string & getField (const std::string &field) const`

Get the value of a field. [More...](#)

`Status getStatus () const`

Get the response status code. [More...](#)

`unsigned int getMajorHttpVersion () const`

Get the major HTTP version number of the response.

`unsigned int getMinorHttpVersion () const`

Get the minor HTTP version number of the response.

`const std::string & getBody () const`

Get the body of the response. [More...](#)

Friends

class **Http**

Detailed Description

Define a HTTP response.

Definition at line 193 of file `Http.hpp`.

Member Enumeration Documentation

enum sf::Http::Response::Status

Enumerate all the valid status codes for a response.

Enumerator	Description
Ok	Most common code returned when operation was successful.
Created	The resource has successfully been created.
Accepted	The request has been accepted, but will be processed later.
NoContent	The server didn't send any data in return.
ResetContent	The server informs the client that it should clear its cache and resend the request to be sent.
PartialContent	The server has sent a part of the resource, a subset of what was requested.
MultipleChoices	The requested page can be accessed from several different URLs.
MovedPermanently	The requested page has permanently moved to a new URL.

MovedTemporarily	The requested page has temporarily moved to another URL.
NotModified	For conditional requests, means the request doesn't need to be refreshed.
BadRequest	The server couldn't understand the request (syntax error).
Unauthorized	The requested page needs an authentication to access.
Forbidden	The requested page cannot be accessed at all, usually due to security reasons.
NotFound	The requested page doesn't exist.
RangeNotSatisfiable	The server can't satisfy the partial GET request.
InternalServerError	The server encountered an unexpected error.
NotImplemented	The server doesn't implement a requested feature.
BadGateway	The gateway server has received an error from another server.
ServiceUnavailable	The server is temporarily unavailable (overloaded or down).
GatewayTimeout	The gateway server couldn't receive a response from another server in time.
VersionNotSupported	The server doesn't support the requested HTTP version.

InvalidResponse	Response is not a valid HTTP one.
ConnectionFailed	Connection with server failed.

Definition at line 201 of file Http.hpp.

Constructor & Destructor Documentation

sf::Http::Response::Response()

Default constructor.

Constructs an empty response.

Member Function Documentation

const std::string& sf::Http::Response::getBody() const

Get the body of the response.

The body of a response may contain:

- the requested page (for GET requests)
- a response from the server (for POST requests)
- nothing (for HEAD requests)
- an error message (in case of an error)

Returns

The response body

const std::string& sf::Http::Response::getField(const std::string &

Get the value of a field.

If the field *field* is not found in the response header, the empty string is returned.
Insensitive comparisons.

Parameters

field Name of the field to get

Returns

Value of the field, or empty string if not found

unsigned int sf::Http::Response::getMajorHttpVersion() const

Get the major HTTP version number of the response.

Returns

Major HTTP version number

See also

[getMinorHttpVersion](#)

unsigned int sf::Http::Response::getMinorHttpVersion() const

Get the minor HTTP version number of the response.

Returns

Minor HTTP version number

See also

[getMajorHttpVersion](#)

Status sf::Http::Response::getStatus() const

Get the response status code.

The status code should be the first thing to be checked after receiving a success, a failure or anything else (see the Status enumeration).

Returns

Status code of the response

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

- [Http.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Member Functions](#) | [Static Public Member Functions](#) | [Static Public Attributes](#) | [Friends](#) | [List of all members](#)

sf::IpAddress Class Reference

[Network module](#)

Encapsulate an IPv4 network address. [More...](#)

#include <IpAddress.hpp>

Public Member Functions

`IpAddress ()`

Default constructor. [More...](#)

`IpAddress (const std::string &address)`

Construct the address from a string. [More...](#)

`IpAddress (const char *address)`

Construct the address from a string. [More...](#)

`IpAddress (Uint8 byte0, Uint8 byte1, Uint8 byte2, Uint8 byte3)`

Construct the address from 4 bytes. [More...](#)

`IpAddress (Uint32 address)`

Construct the address from a 32-bits integer. [More...](#)

`std::string toString () const`

Get a string representation of the address. [More...](#)

`Uint32 toInteger () const`

Get an integer representation of the address. [More...](#)

Static Public Member Functions

static `IpAddress` `getLocalAddress ()`

Get the computer's local address. [More...](#)

static `IpAddress` `getPublicAddress (Time timeout=Time::Zero)`

Get the computer's public address. [More...](#)

Static Public Attributes

static const `IpAddress` `None`

Value representing an empty/invalid address. [More...](#)

static const `IpAddress` `Any`

Value representing any address (0.0.0.0) [More...](#)

static const `IpAddress` `LocalHost`

The "localhost" address (for connecting a computer to itself). [More...](#)

static const `IpAddress` `Broadcast`

The "broadcast" address (for sending UDP messages to all hosts in a network) [More...](#)

Friends

```
bool operator< (const IpAddress &left, const IpAddress &right)  
Overload of < operator to compare two IP addresses. More...
```

Detailed Description

Encapsulate an IPv4 network address.

`sf::IpAddress` is a utility class for manipulating network addresses.

It provides a set of implicit constructors and conversion functions to easily convert from/to various representations.

Usage example:

```
sf::IpAddress a0;                                // an invalid address
sf::IpAddress a1 = sf::IpAddress::None;           // an invalid address
sf::IpAddress a2("127.0.0.1");                  // the local host
sf::IpAddress a3 = sf::IpAddress::Broadcast;      // the broadcast
sf::IpAddress a4(192, 168, 1, 56);                // a local address
sf::IpAddress a5("my_computer");                 // a local address
sf::IpAddress a6("89.54.1.169");                 // a distant address
sf::IpAddress a7("www.google.com");               // a distant address
                                                name
sf::IpAddress a8 = sf::IpAddress::getLocalAddress(); // my address on my local interface
sf::IpAddress a9 = sf::IpAddress::getPublicAddress(); // my address on my public interface
```

Note that `sf::IpAddress` currently doesn't support IPv6 nor other types of networks.

Definition at line 44 of file `IpAddress.hpp`.

Constructor & Destructor Documentation

sf::IpAddress::IpAddress ()

Default constructor.

This constructor creates an empty (invalid) address

sf::IpAddress::IpAddress (const std::string & **address)**

Construct the address from a string.

Here **address** can be either a decimal address (ex: "192.168.1.56") or a name.

Parameters

address IP address or network name

sf::IpAddress::IpAddress (const char * **address)**

Construct the address from a string.

Here **address** can be either a decimal address (ex: "192.168.1.56") or a name.
This is equivalent to the constructor taking a std::string parameter, it is the implicit conversions from literal strings to **IpAddress** work.

Parameters

address IP address or network name

```
sf::IpAddress::IpAddress ( UInt8 byte0,  
                           UInt8 byte1,  
                           UInt8 byte2,  
                           UInt8 byte3  
                         )
```

Construct the address from 4 bytes.

Calling `IpAddress(a, b, c, d)` is equivalent to calling `IpAddress("a.b.c.d")` parse a string to get the address components.

Parameters

- byte0** First byte of the address
- byte1** Second byte of the address
- byte2** Third byte of the address
- byte3** Fourth byte of the address

```
sf::IpAddress::IpAddress ( UInt32 address )
```

Construct the address from a 32-bits integer.

This constructor uses the internal representation of the address for optimization purposes, and only if you got that representation from `IpAddress::toInteger`.

Parameters

- address** 4 bytes of the address packed into a 32-bits integer

See also

`toInteger`

Member Function Documentation

static `IpAddress` `sf::IpAddress::getLocalAddress()`

Get the computer's local address.

The local address is the address of the computer from the LAN port 192.168.1.56. It is meaningful only for communications over the local network; this function is fast and may be used safely anywhere.

Returns

Local IP address of the computer

See also

[getPublicAddress](#)

static `IpAddress` `sf::IpAddress::getPublicAddress(Time timeout = 5s)`

Get the computer's public address.

The public address is the address of the computer from the internet port 89.54.1.169. It is necessary for communications over the world wide web; the address is to ask it to a distant website; as a consequence, this function depends on the network connection and the server, and may be very slow. You should use it a function depends on the network connection and on a distant server, you want your program to be possibly stuck waiting in case there is a problem or default.

Parameters

timeout Maximum time to wait

Returns

Public IP address of the computer

See also

[getLocalAddress](#)

Uint32 sf::IpAddress::toInteger () const

Get an integer representation of the address.

The returned number is the internal representation of the address, and purposes only (like sending the address through a socket). The integer can then be converted back to a [sf::IpAddress](#) with the proper constructor.

Returns

32-bits unsigned integer representation of the address

See also

[toString](#)

std::string sf::IpAddress::toString () const

Get a string representation of the address.

The returned string is the decimal representation of the IP address (like "192.168.1.1"), or it can be constructed from a host name.

Returns

[String](#) representation of the address

See also

[toInteger](#)

Friends And Related Function Documentation

```
bool operator< ( const IpAddress & left,  
                  const IpAddress & right  
                )
```

Overload of < operator to compare two IP addresses.

Parameters

left Left operand (a IP address)
right Right operand (a IP address)

Returns

True if *left* is lesser than *right*

Member Data Documentation

const `IpAddress` `sf::IpAddress::Any`

Value representing any address (0.0.0.0)

Definition at line [185](#) of file `IpAddress.hpp`.

const `IpAddress` `sf::IpAddress::Broadcast`

The "broadcast" address (for sending UDP messages to everyone on a local network)

Definition at line [187](#) of file `IpAddress.hpp`.

const `IpAddress` `sf::IpAddress::LocalHost`

The "localhost" address (for connecting a computer to itself locally)

Definition at line [186](#) of file `IpAddress.hpp`.

const `IpAddress` `sf::IpAddress::None`

Value representing an empty/invalid address.

Definition at line [184](#) of file `IpAddress.hpp`.

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

- [IpAddress.hpp](#)
-

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Protected Member Functions](#) | [Friends](#) | [List of all members](#)

sf::Packet Class Reference

[Network module](#)

Utility class to build blocks of data to transfer over the network. More...

```
#include <Packet.hpp>
```

Public Member Functions

[Packet \(\)](#)

Default constructor. [More...](#)

`virtual ~Packet ()`

Virtual destructor. [More...](#)

`void append (const void *data, std::size_t sizeInBytes)`

Append data to the end of the packet. [More...](#)

`void clear ()`

Clear the packet. [More...](#)

`const void * getData () const`

Get a pointer to the data contained in the packet. [More...](#)

`std::size_t getDataSize () const`

Get the size of the data contained in the packet. [More...](#)

`bool endOfPacket () const`

Tell if the reading position has reached the end of the packe

`operator BoolType () const`

Test the validity of the packet, for reading. [More...](#)

`Packet & operator>> (bool &data)`

Overloads of operator >> to read data from the packet. [More...](#)

`Packet & operator>> (Int8 &data)`

`Packet & operator>> (Uint8 &data)`

`Packet & operator>> (Int16 &data)`

`Packet & operator>> (Uint16 &data)`

`Packet & operator>> (Int32 &data)`

`Packet & operator>> (Uint32 &data)`

`Packet & operator>> (Int64 &data)`

`Packet & operator>> (Uint64 &data)`

`Packet & operator>> (float &data)`

`Packet & operator>> (double &data)`

`Packet & operator>> (char *data)`

`Packet & operator>> (std::string &data)`

`Packet & operator>> (wchar_t *data)`

`Packet & operator>> (std::wstring &data)`

`Packet & operator>> (String &data)`

`Packet & operator<< (bool data)`

Overloads of operator << to write data into the packet. More

`Packet & operator<< (Int8 data)`

`Packet & operator<< (Uint8 data)`

`Packet & operator<< (Int16 data)`

`Packet & operator<< (Uint16 data)`

Packet & **operator<<** (Int32 data)

Packet & **operator<<** (Uint32 data)

Packet & **operator<<** (Int64 data)

Packet & **operator<<** (Uint64 data)

Packet & **operator<<** (float data)

Packet & **operator<<** (double data)

Packet & **operator<<** (const char *data)

Packet & **operator<<** (const std::string &data)

Packet & **operator<<** (const wchar_t *data)

Packet & **operator<<** (const std::wstring &data)

Packet & **operator<<** (const String &data)

Protected Member Functions

`virtual const void * onSend (std::size_t &size)`

Called before the packet is sent over the network. [More...](#)

`virtual void onReceive (const void *data, std::size_t size)`

Called after the packet is received over the network.

Friends

class **TcpSocket**

class **UdpSocket**

Detailed Description

Utility class to build blocks of data to transfer over the network.

Packets provide a safe and easy way to serialize data, in order to send (sf::TcpSocket, sf::UdpSocket).

Packets solve 2 fundamental problems that arise when transferring data over the network:

- data is interpreted correctly according to the endianness
- the bounds of the packet are preserved (one send == one receive)

The sf::Packet class provides both input and output modes. It is designed to be used with C++ streams, using operators >> and << to extract and insert data.

It is recommended to use only fixed-size types (like sf::Int32, etc.), to avoid alignment issues between the sender and the receiver. Indeed, the native C++ types may have different sizes on different platforms, so your data may be corrupted if that happens.

Usage example:

```
sf::Uint32 x = 24;
std::string s = "hello";
double d = 5.89;

// Group the variables to send into a packet
sf::Packet packet;
packet << x << s << d;

// Send it over the network (socket is a valid sf::TcpSocket)
socket.send(packet);

-----
// Receive the packet at the other end
sf::Packet packet;
socket.receive(packet);
```

```
// Extract the variables contained in the packet
sf::Uint32 x;
std::string s;
double d;
if (packet >> x >> s >> d)
{
    // Data extracted successfully...
}
```

Packets have built-in operator >> and << overloads for standard types:

- bool
- fixed-size integer types (sf::Int8/16/32, sf::Uint8/16/32)
- floating point numbers (float, double)
- string types (char*, wchar_t*, std::string, std::wstring, sf::String)

Like standard streams, it is also possible to define your own overloads to handle your custom types.

```
struct MyStruct
{
    float      number;
    sf::Int8    integer;
    std::string str;
};

sf::Packet& operator <<(sf::Packet& packet, const MyStruct& m)
{
    return packet << m.number << m.integer << m.str;
}

sf::Packet& operator >>(sf::Packet& packet, MyStruct& m)
{
    return packet >> m.number >> m.integer >> m.str;
}
```

Packets also provide an extra feature that allows to apply custom transformations, before it is sent, and after it is received. This is typically used to handle automatic data. This is achieved by inheriting from **sf::Packet**, and overriding the onSend and onReceive methods.

Here is an example:

```
class ZipPacket : public sf::Packet
{
    virtual const void* onSend(std::size_t& size)
    {
        const void* srcData = getData();
        std::size_t srcSize = getDataSize();

        return MySuperZipFunction(srcData, srcSize, &size);
    }

    virtual void onReceive(const void* data, std::size_t size)
    {
        std::size_t dstSize;
        const void* dstData = MySuperUnzipFunction(data, size, &dstSize);

        append(dstData, dstSize);
    }
};

// Use like regular packets:
ZipPacket packet;
packet << x << s << d;
...
```

See also

[sf::TcpSocket](#), [sf::UdpSocket](#)

Definition at line 47 of file [Packet.hpp](#).

Constructor & Destructor Documentation

sf::Packet::Packet()

Default constructor.

Creates an empty packet.

virtual sf::Packet::~Packet()

Virtual destructor.

Member Function Documentation

```
void sf::Packet::append ( const void * data,  
                         std::size_t sizeInBytes  
                     )
```

Append data to the end of the packet.

Parameters

data Pointer to the sequence of bytes to append
sizeInBytes Number of bytes to append

See also

[clear](#)

```
void sf::Packet::clear ( )
```

Clear the packet.

After calling Clear, the packet is empty.

See also

[append](#)

```
bool sf::Packet::endOfPacket ( ) const
```

Tell if the reading position has reached the end of the packet.

This function is useful to know if there is some data left to be read, without having to read it.

Returns

True if all data was read, false otherwise

See also

[operator bool](#)

const void* sf::Packet::getData () const

Get a pointer to the data contained in the packet.

Warning: the returned pointer may become invalid after you append or modify the data. This pointer should never be stored. The return pointer is NULL if the packet is empty.

Returns

Pointer to the data

See also

[getDataSize](#)

std::size_t sf::Packet::getDataSize () const

Get the size of the data contained in the packet.

This function returns the number of bytes pointed to by what getData returns.

Returns

Data size, in bytes

See also

[getData](#)

```
virtual void sf::Packet::onReceive ( const void * data,
                                    std::size_t    size
                                )
```

Called after the packet is received over the network.

This function can be defined by derived classes to transform the data used for decompression, decryption, etc. The function receives a pointer to the packet and must fill the packet with the transformed bytes. The default implementation does nothing, leaving the data unchanged.

Parameters

data Pointer to the received bytes

size Number of bytes

See also

[onSend](#)

```
virtual const void* sf::Packet::onSend ( std::size_t & size )
```

Called before the packet is sent over the network.

This function can be defined by derived classes to transform the data before it is sent over the network. The function must return a pointer to the transformed data and update the size parameter to reflect the number of bytes pointed to. The default implementation provides the packet as-is.

Parameters

size Variable to fill with the size of data to send

Returns

Pointer to the array of bytes to send

See also

[onReceive](#)

sf::Packet::operator BoolType () const

Test the validity of the packet, for reading.

This operator allows to test the packet as a boolean variable, to check if the read was successful.

A packet will be in an invalid state if it has no more data to read.

This behavior is the same as standard C++ streams.

Usage example:

```
float x;
packet >> x;
if (packet)
{
    // ok, x was extracted successfully
}

// -- or --

float x;
if (packet >> x)
{
    // ok, x was extracted successfully
}
```

Don't focus on the return type, it's equivalent to bool but it disallows `>>`

integer or pointer types.

Returns

True if last data extraction from packet was successful

See also

[endOfPacket](#)

Packet& sf::Packet::operator<< (bool data)

Overloads of operator << to write data into the packet.

Packet& sf::Packet::operator>> (bool & data)

Overloads of operator >> to read data from the packet.

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

- [Packet.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [Protected Types](#) | [Protected Member Functions](#) | [Friends](#) | [List of all members](#)

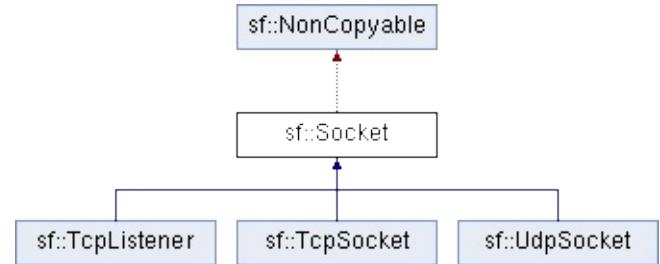
sf::Socket Class Reference

Network module

Base class for all the socket types. More...

```
#include <Socket.hpp>
```

Inheritance diagram for sf::Socket:



Public Types

```
enum Status {  
    Done, NotReady, Partial, Disconnected,  
    Error  
}
```

Status codes that may be returned by socket functions. [More...](#)

```
enum { AnyPort = 0 }
```

Some special values used by sockets. [More...](#)

Public Member Functions

`virtual ~Socket ()`

Destructor. [More...](#)

`void setBlocking (bool blocking)`

Set the blocking state of the socket. [More...](#)

`bool isBlocking () const`

Tell whether the socket is in blocking or non-blocking mode. [More...](#)

Protected Types

enum **Type** { **Tcp**, **Udp** }

Types of protocols that the socket can use. [More...](#)

Protected Member Functions

`Socket (Type type)`

Default constructor. [More...](#)

`SocketHandle getHandle () const`

Return the internal handle of the socket. [More...](#)

`void create ()`

Create the internal representation of the socket. [More...](#)

`void create (SocketHandle handle)`

Create the internal representation of the socket from a so

`void close ()`

Close the socket gracefully. [More...](#)

Friends

```
class SocketSelector
```

Detailed Description

Base class for all the socket types.

This class mainly defines internal stuff to be used by derived classes.

The only public features that it defines, and which is therefore common to all derived classes, is the blocking state. All sockets can be set as blocking or non-blocking.

In blocking mode, socket functions will hang until the operation completes. This means that the program (well, in fact the current thread if you use multiple ones) will wait for the operation to complete.

In non-blocking mode, all the socket functions will return immediately. If the requested operation is not yet completed, the function simply returns the proper status code.

The default mode, which is blocking, is the one that is generally used for most applications. The non-blocking mode is rather used in real-time applications where the application needs to poll the socket often enough, and cannot afford blocking this loop.

See also

[sf::TcpListener](#), [sf::TcpSocket](#), [sf::UdpSocket](#)

Definition at line 45 of file [Socket.hpp](#).

Member Enumeration Documentation

anonymous enum

Some special values used by sockets.

Enumerator

AnyPort

Special value that tells the system to pick any available port.

Definition at line 66 of file [Socket.hpp](#).

enum sf::Socket::Status

Status codes that may be returned by socket functions.

Enumerator

Done

The socket has sent / received the data.

NotReady

The socket is not ready to send / receive data yet.

Partial

The socket sent a part of the data.

Disconnected

The TCP socket has been disconnected.

Error	An unexpected error happened.
-------	-------------------------------

Definition at line 53 of file [Socket.hpp](#).

enum sf::Socket::Type

Types of protocols that the socket can use.

Enumerator

Tcp

TCP protocol.

Udp

UDP protocol.

Definition at line 114 of file [Socket.hpp](#).

Constructor & Destructor Documentation

virtual sf::Socket::~Socket()

Destructor.

sf::Socket::Socket(Type type)

Default constructor.

This constructor can only be accessed by derived classes.

Parameters

type Type of the socket (TCP or UDP)

Member Function Documentation

void sf::Socket::close()

Close the socket gracefully.

This function can only be accessed by derived classes.

void sf::Socket::create()

Create the internal representation of the socket.

This function can only be accessed by derived classes.

void sf::Socket::create(SocketHandle handle)

Create the internal representation of the socket from a socket handle.

This function can only be accessed by derived classes.

Parameters

handle OS-specific handle of the socket to wrap

SocketHandle sf::Socket::getHandle() const

Return the internal handle of the socket.

The returned handle may be invalid if the socket was not created by this function, and the function can only be accessed by derived classes.

Returns

The internal (OS-specific) handle of the socket

bool sf::Socket::isBlocking () const

Tell whether the socket is in blocking or non-blocking mode.

Returns

True if the socket is blocking, false otherwise

See also

[setBlocking](#)

void sf::Socket::setBlocking (bool **blocking)**

Set the blocking state of the socket.

In blocking mode, calls will not return until they have completed their task. In non-blocking mode won't return until some data was actually received. Non-blocking sockets always return immediately, using the return code to signal whether the operation succeeded or failed. Note that by default, all sockets are blocking.

Parameters

blocking True to set the socket as blocking, false for non-blocking

See also

[isBlocking](#)

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

- [Socket.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [List of all members](#)

sf::SocketSelector Class Reference

[Network module](#)

Multiplexer that allows to read from multiple sockets. [More...](#)

```
#include <SocketSelector.hpp>
```

Public Member Functions

`SocketSelector ()`

Default constructor. [More...](#)

`SocketSelector (const SocketSelector ©)`

Copy constructor. [More...](#)

`~SocketSelector ()`

Destructor. [More...](#)

`void add (Socket &socket)`

Add a new socket to the selector. [More...](#)

`void remove (Socket &socket)`

Remove a socket from the selector. [More...](#)

`void clear ()`

Remove all the sockets stored in the selector. [More...](#)

`bool wait (Time timeout=Time::Zero)`

Wait until one or more sockets are ready to receive. [More...](#)

`bool isReady (Socket &socket) const`

Test a socket to know if it is ready to receive data. [More...](#)

`SocketSelector & operator= (const SocketSelector &right)`

Overload of assignment operator. [More...](#)

Detailed Description

Multiplexer that allows to read from multiple sockets.

Socket selectors provide a way to wait until some data is available on a set of sockets.

This is convenient when you have multiple sockets that may possibly be ready at different times, and you want to know which one will be ready first. In particular, it avoids to use a thread for each socket, because a single thread can handle all the sockets.

All types of sockets can be used in a selector:

- sf::TcpListener
- sf::TcpSocket
- sf::UdpSocket

A selector doesn't store its own copies of the sockets (socket classes are reference-counted), so it only keeps a reference to the original sockets that you pass to the "add" function. If you want to use a selector as a socket container, you must store them outside and make sure they are copied to the selector when they are used in the selector.

Using a selector is simple:

- populate the selector with all the sockets that you want to observe
- make it wait until there is data available on any of the sockets
- test each socket to find out which ones are ready

Usage example:

```
// Create a socket to listen to new connections
sf::TcpListener listener;
listener.listen(55001);
```

```

// Create a list to store the future clients
std::list<sf::TcpSocket*> clients;

// Create a selector
sf::SocketSelector selector;

// Add the listener to the selector
selector.add(listener);

// Endless loop that waits for new connections
while (running)
{
    // Make the selector wait for data on any socket
    if (selector.wait())
    {
        // Test the listener
        if (selector.isReady(listener))
        {
            // The listener is ready: there is a pending connection
            sf::TcpSocket* client = new sf::TcpSocket;
            if (listener.accept(*client) == sf::Socket::Done)
            {
                // Add the new client to the clients list
                clients.push_back(client);

                // Add the new client to the selector so that we will
                // be notified when he sends something
                selector.add(*client);
            }
        }
        else
        {
            // Error, we won't get a new connection, delete the socket
            delete client;
        }
    }
    else
    {
        // The listener socket is not ready, test all other sockets (the client)
        for (std::list<sf::TcpSocket*>::iterator it = clients.begin(); it != clients.end();)
        {
            sf::TcpSocket& client = **it;
            if (selector.isReady(client))
            {
                // The client has sent some data, we can receive it
                sf::Packet packet;
                if (client.receive(packet) == sf::Socket::Done)
                {
                    ...
                }
            }
        }
    }
}

```

```
}
```

See also

[sf::Socket](#)

Definition at line 43 of file [SocketSelector.hpp](#).

Constructor & Destructor Documentation

`sf::SocketSelector::SocketSelector()`

Default constructor.

`sf::SocketSelector::SocketSelector(const SocketSelector & copy)`

Copy constructor.

Parameters

copy Instance to copy

`sf::SocketSelector::~SocketSelector()`

Destructor.

Member Function Documentation

void sf::SocketSelector::add (Socket & socket)

Add a new socket to the selector.

This function keeps a weak reference to the socket, so you have to make sure it's not destroyed while it is stored in the selector. This function does nothing if the socket is already added.

Parameters

socket Reference to the socket to add

See also

[remove](#), [clear](#)

void sf::SocketSelector::clear ()

Remove all the sockets stored in the selector.

This function doesn't destroy any instance, it simply removes all the references to external sockets.

See also

[add](#), [remove](#)

bool sf::SocketSelector::isReady (Socket & socket) const

Test a socket to know if it is ready to receive data.

This function must be used after a call to Wait, to know which sockets' socket is ready, a call to receive will never block because we know that Note that if this function returns true for a `TcpListener`, this means that connection.

Parameters

socket `Socket` to test

Returns

True if the socket is ready to read, false otherwise

See also

`isReady`

`SocketSelector& sf::SocketSelector::operator= (const SocketSelecto`

Overload of assignment operator.

Parameters

right Instance to assign

Returns

Reference to self

`void sf::SocketSelector::remove (Socket & socket)`

Remove a socket from the selector.

This function doesn't destroy the socket, it simply removes the reference

Parameters

socket Reference to the socket to remove

See also

[add](#), [clear](#)

bool sf::SocketSelector::wait (Time timeout = Time::Zero)

Wait until one or more sockets are ready to receive.

This function returns as soon as at least one socket has some data available. To know which sockets are ready, use the [isReady](#) function. If you use a timeout of zero, the function returns false.

Parameters

timeout Maximum time to wait, (use [Time::Zero](#) for infinity)

Returns

True if there are sockets ready, false otherwise

See also

[isReady](#)

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

- [SocketSelector.hpp](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [Protected Types](#) | [Protected Member Functions](#) | [List of all members](#)

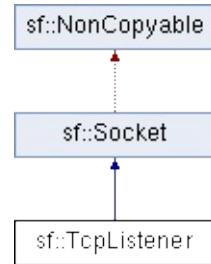
sf::TcpListener Class Reference

Network module

Socket that listens to new TCP connections. [More...](#)

```
#include <TcpListener.hpp>
```

Inheritance diagram for sf::TcpListener:



Public Types

```
enum Status {  
    Done, NotReady, Partial, Disconnected,  
    Error  
}
```

Status codes that may be returned by socket functions. [More...](#)

```
enum { AnyPort = 0 }
```

Some special values used by sockets. [More...](#)

Public Member Functions

`TcpListener ()`

Default constructor. [More...](#)

`unsigned short getLocalPort () const`

Get the port to which the socket is bound locally. [More...](#)

`Status listen (unsigned short port, const IpAddress &address=I`

Start listening for connections. [More...](#)

`void close ()`

Stop listening and close the socket. [More...](#)

`Status accept (TcpSocket &socket)`

Accept a new connection. [More...](#)

`void setBlocking (bool blocking)`

Set the blocking state of the socket. [More...](#)

`bool isBlocking () const`

Tell whether the socket is in blocking or non-blocking mo

Protected Types

enum **Type** { **Tcp**, **Udp** }

Types of protocols that the socket can use. [More...](#)

Protected Member Functions

SocketHandle **getHandle () const**

Return the internal handle of the socket. [More...](#)

void **create ()**

Create the internal representation of the socket. [More...](#)

void **create (SocketHandle handle)**

Create the internal representation of the socket from a so

Detailed Description

Socket that listens to new TCP connections.

A listener socket is a special type of socket that listens to a given port and accepts connections on that port.

This is all it can do.

When a new connection is received, you must call `accept` and the listener will return a `sf::TcpSocket` that is properly initialized and can be used to communicate with the client.

Listener sockets are specific to the TCP protocol, UDP sockets are completely different and cannot be used to communicate directly. As a consequence, a listener socket will always return `sf::TcpSocket` instances.

A listener is automatically closed on destruction, like all other types of sockets. If you want to stop listening before the socket is destroyed, you can call its `close()` function.

Usage example:

```
// Create a listener socket and make it wait for new
// connections on port 55001
sf::TcpListener listener;
listener.listen(55001);

// Endless loop that waits for new connections
while (running)
{
    sf::TcpSocket client;
    if (listener.accept(client) == sf::Socket::Done)
    {
        // A new client just connected!
        std::cout << "New connection received from " << client.getRemoteAddress();
        doSomethingWith(client);
    }
}
```

See also

[sf::TcpSocket](#), [sf::Socket](#)

Definition at line 44 of file [TcpListener.hpp](#).

Member Enumeration Documentation

anonymous enum

Some special values used by sockets.

Enumerator

AnyPort

Special value that tells the system to pick any available port.

Definition at line 66 of file [Socket.hpp](#).

enum sf::Socket::Status

Status codes that may be returned by socket functions.

Enumerator

Done

The socket has sent / received the data.

NotReady

The socket is not ready to send / receive data yet.

Partial

The socket sent a part of the data.

Disconnected

The TCP socket has been disconnected.

Error	An unexpected error happened.
-------	-------------------------------

Definition at line 53 of file [Socket.hpp](#).

enum sf::Socket::Type

Types of protocols that the socket can use.

Enumerator

Tcp	TCP protocol.
Udp	UDP protocol.

Definition at line 114 of file [Socket.hpp](#).

Constructor & Destructor Documentation

sf::TcpListener::TcpListener()

Default constructor.

Member Function Documentation

Status sf::TcpListener::accept (TcpSocket & **socket**)

Accept a new connection.

If the socket is in blocking mode, this function will not return until a connection is accepted.

Parameters

socket Socket that will hold the new connection

Returns

Status code

See also

[listen](#)

void sf::TcpListener::close ()

Stop listening and close the socket.

This function gracefully stops the listener. If the socket is not listening, the function does nothing.

See also

[listen](#)

void sf::Socket::create ()

Create the internal representation of the socket.

This function can only be accessed by derived classes.

void sf::Socket::create (SocketHandle handle)

Create the internal representation of the socket from a socket handle.

This function can only be accessed by derived classes.

Parameters

handle OS-specific handle of the socket to wrap

SocketHandle sf::Socket::getHandle () const

Return the internal handle of the socket.

The returned handle may be invalid if the socket was not created
function can only be accessed by derived classes.

Returns

The internal (OS-specific) handle of the socket

unsigned short sf::TcpListener::getLocalPort () const

Get the port to which the socket is bound locally.

If the socket is not listening to a port, this function returns 0.

Returns

Port to which the socket is bound

See also

[listen](#)

bool sf::Socket::isBlocking () const

Tell whether the socket is in blocking or non-blocking mode.

Returns

True if the socket is blocking, false otherwise

See also

[setBlocking](#)

Status sf::TcpListener::listen (unsigned short port, const IpAddress & address = IpAddress::any())

Start listening for connections.

This function makes the socket listen to the specified port, waiting for a connection. If the socket was previously listening to another port, it will be stopped first and bound to the new port.

Parameters

port Port to listen for new connections

address Address of the interface to listen on

Returns

Status code

See also

[accept](#), [close](#)

void sf::Socket::setBlocking (bool **blocking)**

Set the blocking state of the socket.

In blocking mode, calls will not return until they have completed their task. In non-blocking mode, calls will return immediately, using the return code to signal whether the operation was successful or not. By default, all sockets are blocking.

Parameters

blocking True to set the socket as blocking, false for non-blocking

See also

[isBlocking](#)

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

- [TcpListener.hpp](#)

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Classes](#) | [Public Types](#) | [Public Member Functions](#) | [Protected Types](#) | [Protected Member Functions](#) | [Friends](#) | [List of all members](#)

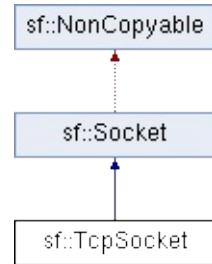
sf::TcpSocket Class Reference

Network module

Specialized socket using the TCP protocol. [More...](#)

```
#include <TcpSocket.hpp>
```

Inheritance diagram for sf::TcpSocket:



Public Types

```
enum Status {  
    Done, NotReady, Partial, Disconnected,  
    Error  
}
```

Status codes that may be returned by socket functions. [More...](#)

```
enum { AnyPort = 0 }
```

Some special values used by sockets. [More...](#)

Public Member Functions

`TcpSocket ()`
Default constructor. [More...](#)

`unsigned short getLocalPort () const`
Get the port to which the socket is bound locally. [More...](#)

`IpAddress getRemoteAddress () const`
Get the address of the connected peer. [More...](#)

`unsigned short getRemotePort () const`
Get the port of the connected peer to which the socket is

`Status connect (const IpAddress &remoteAddress, unsigned short timeout=Time::Zero)`
Connect the socket to a remote peer. [More...](#)

`void disconnect ()`
Disconnect the socket from its remote peer. [More...](#)

`Status send (const void *data, std::size_t size)`
Send raw data to the remote peer. [More...](#)

`Status send (const void *data, std::size_t size, std::size_t &sent)`
Send raw data to the remote peer. [More...](#)

`Status receive (void *data, std::size_t size, std::size_t &received)`
Receive raw data from the remote peer. [More...](#)

`Status send (Packet &packet)`
Send a formatted packet of data to the remote peer. [More...](#)

Status receive (Packet &packet)

Receive a formatted packet of data from the remote peer.

void setBlocking (bool blocking)

Set the blocking state of the socket. [More...](#)

bool isBlocking () const

Tell whether the socket is in blocking or non-blocking mode.

Protected Types

enum **Type** { **Tcp**, **Udp** }

Types of protocols that the socket can use. [More...](#)

Protected Member Functions

SocketHandle **getHandle () const**

Return the internal handle of the socket. [More...](#)

void **create ()**

Create the internal representation of the socket. [More...](#)

void **create (SocketHandle handle)**

Create the internal representation of the socket from a so

void **close ()**

Close the socket gracefully. [More...](#)

Friends

```
class TcpListener
```

Detailed Description

Specialized socket using the TCP protocol.

TCP is a connected protocol, which means that a TCP socket can only be connected to.

It can't send or receive anything if it is not connected.

The TCP protocol is reliable but adds a slight overhead. It ensures that your data is sent in order and without errors (no data corrupted, lost or duplicated).

When a socket is connected to a remote host, you can retrieve information about the connection with the getRemoteAddress and getRemotePort functions. You can also get the local port bound (which is automatically chosen when the socket is connected), with the getLocalPort function.

Sending and receiving data can use either the low-level or the high-level interface. The low-level interface allows you to directly process a raw sequence of bytes, and cannot ensure that one call to `Send` corresponds to one call to `Receive` at the other end of the socket.

The high-level interface uses packets (see [sf::Packet](#)), which are easier to work with regarding the data that is exchanged. You can look at the [sf::Packet](#) class documentation to learn more about how they work.

The socket is automatically disconnected when it is destroyed, but if you want to disconnect the connection while the socket instance is still alive, you can call `disconnect`.

Usage example:

```
// ----- The client -----
// Create a socket and connect it to 192.168.1.50 on port 55001
sf::TcpSocket socket;
```

```
socket.connect("192.168.1.50", 55001);

// Send a message to the connected host
std::string message = "Hi, I am a client";
socket.send(message.c_str(), message.size() + 1);

// Receive an answer from the server
char buffer[1024];
std::size_t received = 0;
socket.receive(buffer, sizeof(buffer), received);
std::cout << "The server said: " << buffer << std::endl;

// ----- The server -----

// Create a listener to wait for incoming connections on port 55001
sf::TcpListener listener;
listener.listen(55001);

// Wait for a connection
sf::TcpSocket socket;
listener.accept(socket);
std::cout << "New client connected: " << socket.getRemoteAddress() << std::endl;

// Receive a message from the client
char buffer[1024];
std::size_t received = 0;
socket.receive(buffer, sizeof(buffer), received);
std::cout << "The client said: " << buffer << std::endl;

// Send an answer
std::string message = "Welcome, client";
socket.send(message.c_str(), message.size() + 1);
```

See also

[sf::Socket](#), [sf::UdpSocket](#), [sf::Packet](#)

Definition at line 46 of file [TcpSocket.hpp](#).

Member Enumeration Documentation

anonymous enum

Some special values used by sockets.

Enumerator

AnyPort

Special value that tells the system to pick any available port.

Definition at line 66 of file [Socket.hpp](#).

enum sf::Socket::Status

Status codes that may be returned by socket functions.

Enumerator

Done

The socket has sent / received the data.

NotReady

The socket is not ready to send / receive data yet.

Partial

The socket sent a part of the data.

Disconnected

The TCP socket has been disconnected.

Error	An unexpected error happened.
-------	-------------------------------

Definition at line 53 of file [Socket.hpp](#).

enum sf::Socket::Type

Types of protocols that the socket can use.

Enumerator

Tcp	TCP protocol.
Udp	UDP protocol.

Definition at line 114 of file [Socket.hpp](#).

Constructor & Destructor Documentation

sf::TcpSocket::TcpSocket()

Default constructor.

Member Function Documentation

void sf::Socket::close()

Close the socket gracefully.

This function can only be accessed by derived classes.

```
Status sf::TcpSocket::connect( const IpAddress & remoteAddress
                               unsigned short remotePort,
                               Time timeout = Time::  
)
```

Connect the socket to a remote peer.

In blocking mode, this function may take a while, especially if the **remoteAddress** parameter allows you to stop trying to connect after a given timeout. If the connection fails, it is first disconnected.

Parameters

remoteAddress	Address of the remote peer
remotePort	Port of the remote peer
timeout	Optional maximum time to wait

Returns

Status code

See also

[disconnect](#)

void sf::Socket::create()

Create the internal representation of the socket.

This function can only be accessed by derived classes.

void sf::Socket::create(SocketHandle handle)

Create the internal representation of the socket from a socket handle.

This function can only be accessed by derived classes.

Parameters

handle OS-specific handle of the socket to wrap

void sf::TcpSocket::disconnect()

Disconnect the socket from its remote peer.

This function gracefully closes the connection. If the socket is not connected, it has no effect.

See also

[connect](#)

SocketHandle sf::Socket::getHandle() const

Return the internal handle of the socket.

The returned handle may be invalid if the socket was not created by this function. This function can only be accessed by derived classes.

Returns

The internal (OS-specific) handle of the socket

unsigned short sf::TcpSocket::getLocalPort() const

Get the port to which the socket is bound locally.

If the socket is not connected, this function returns 0.

Returns

Port to which the socket is bound

See also

[connect](#), [getRemotePort](#)

IpAddress sf::TcpSocket::getRemoteAddress() const

Get the address of the connected peer.

If the socket is not connected, this function returns [sf::IpAddress::None](#).

Returns

Address of the remote peer

See also

getRemotePort

unsigned short sf::TcpSocket::getRemotePort() const

Get the port of the connected peer to which the socket is connected.

If the socket is not connected, this function returns 0.

Returns

Remote port to which the socket is connected

See also

[getRemoteAddress](#)

bool sf::Socket::isBlocking() const

Tell whether the socket is in blocking or non-blocking mode.

Returns

True if the socket is blocking, false otherwise

See also

[setBlocking](#)

```
Status sf::TcpSocket::receive( void *      data,
                               std::size_t    size,
                               std::size_t & received
                           )
```

Receive raw data from the remote peer.

In blocking mode, this function will wait until some bytes are actually received. If the socket is not connected.

Parameters

- data** Pointer to the array to fill with the received bytes
- size** Maximum number of bytes that can be received
- received** This variable is filled with the actual number of bytes received

Returns

Status code

See also

[send](#)

Status `sf::TcpSocket::receive (Packet & packet)`

Receive a formatted packet of data from the remote peer.

In blocking mode, this function will wait until the whole packet has been received. If the socket is not connected.

Parameters

- packet** Packet to fill with the received data

Returns

Status code

See also

[send](#)

```
Status sf::TcpSocket::send ( const void * data,
                            std::size_t    size
                          )
```

Send raw data to the remote peer.

To be able to handle partial sends over non-blocking sockets, use the `std::size_t&` overload instead. This function will fail if the socket is not connected.

Parameters

data Pointer to the sequence of bytes to send
size Number of bytes to send

Returns

Status code

See also

[receive](#)

```
Status sf::TcpSocket::send ( const void * data,
                            std::size_t    size,
                            std::size_t & sent
                          )
```

Send raw data to the remote peer.

This function will fail if the socket is not connected.

Parameters

data Pointer to the sequence of bytes to send
size Number of bytes to send

sent The number of bytes sent will be written here

Returns

Status code

See also

[receive](#)

Status `sf::TcpSocket::send (Packet & packet)`

Send a formatted packet of data to the remote peer.

In non-blocking mode, if this function returns `sf::Socket::Partial`, you unmodified packet before sending anything else in order to guarantee the peer uncorrupted. This function will fail if the socket is not connected.

Parameters

packet Packet to send

Returns

Status code

See also

[receive](#)

`void sf::Socket::setBlocking (bool blocking)`

Set the blocking state of the socket.

In blocking mode, calls will not return until they have completed their task. In non-blocking mode won't return until some data was actually received.

always return immediately, using the return code to signal whether the default, all sockets are blocking.

Parameters

blocking True to set the socket as blocking, false for non-blocking

See also

[isBlocking](#)

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

- [TcpSocket.hpp](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [Protected Types](#) | [Protected Member Functions](#) | [List of all members](#)

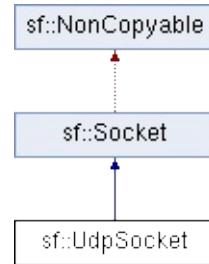
sf::UdpSocket Class Reference

[Network module](#)

Specialized socket using the UDP protocol. [More...](#)

```
#include <UdpSocket.hpp>
```

Inheritance diagram for sf::UdpSocket:



Public Types

enum { **MaxDatagramSize** = 65507 }

```
enum Status {
    Done, NotReady, Partial, Disconnected,
    Error
}
```

Status codes that may be returned by socket functions. [More...](#)

enum { **AnyPort** = 0 }

Some special values used by sockets. [More...](#)

Public Member Functions

[UdpSocket \(\)](#)

Default constructor. [More...](#)

unsigned short [getLocalPort \(\) const](#)

Get the port to which the socket is bound locally. [More...](#)

Status [bind \(unsigned short port, const IpAddress &address=Ip\)](#)

Bind the socket to a specific port. [More...](#)

void [unbind \(\)](#)

Unbind the socket from the local port to which it is bound.

Status [send \(const void *data, std::size_t size, const IpAddress &remotePort\)](#)

Send raw data to a remote peer. [More...](#)

Status [receive \(void *data, std::size_t size, std::size_t &received, unsigned short &remotePort\)](#)

Receive raw data from a remote peer. [More...](#)

Status [send \(Packet &packet, const IpAddress &remoteAddress\)](#)

Send a formatted packet of data to a remote peer. [More...](#)

Status [receive \(Packet &packet, IpAddress &remoteAddress, unsigned short &remotePort\)](#)

Receive a formatted packet of data from a remote peer. [More...](#)

void [setBlocking \(bool blocking\)](#)

Set the blocking state of the socket. [More...](#)

bool [isBlocking \(\) const](#)

Tell whether the socket is in blocking or non-blocking mode.

Protected Types

enum **Type** { **Tcp**, **Udp** }

Types of protocols that the socket can use. [More...](#)

Protected Member Functions

SocketHandle **getHandle () const**

Return the internal handle of the socket. [More...](#)

void **create ()**

Create the internal representation of the socket. [More...](#)

void **create (SocketHandle handle)**

Create the internal representation of the socket from a so

void **close ()**

Close the socket gracefully. [More...](#)

Detailed Description

Specialized socket using the UDP protocol.

A UDP socket is a connectionless socket.

Instead of connecting once to a remote host, like TCP sockets, it can serve any time.

It is a datagram protocol: bounded blocks of data (datagrams) are transferred, not a continuous stream of data (TCP). Therefore, one call to send will always return when the datagram is not lost), with the same data that was sent.

The UDP protocol is lightweight but unreliable. Unreliable means that data may be lost or arrive reordered. However, if a datagram arrives, its data is guaranteed to be correct.

UDP is generally used for real-time communication (audio or video streams). Speed is crucial and lost data doesn't matter much.

Sending and receiving data can use either the low-level or the high-level interface. The low-level interface processes a raw sequence of bytes, whereas the high-level interface uses `sf::Packet` objects. These are easier to use and provide more safety regarding the data that is transferred. See the `sf::Packet` class to get more details about how they work.

It is important to note that `UdpSocket` is unable to send datagrams bigger than 512 bytes. In such case, it returns an error and doesn't send anything. This applies to both sending and receiving. Even though packets are unable to split and recompose data, due to the unreliable nature of the protocol, mixing or duplicated datagrams may lead to a big mess when trying to reconstruct the original message.

If the socket is bound to a port, it is automatically unbound from it when destroyed. If you need to unbind the socket explicitly with the `Unbind` function if necessary.

make the port available for other sockets.

Usage example:

```
// ----- The client -----
// Create a socket and bind it to the port 55001
sf::UdpSocket socket;
socket.bind(55001);

// Send a message to 192.168.1.50 on port 55002
std::string message = "Hi, I am " + sf::IpAddress::getLocalAddress().toString();
socket.send(message.c_str(), message.size() + 1, "192.168.1.50", 55002);

// Receive an answer (most likely from 192.168.1.50, but could be anyone)
char buffer[1024];
std::size_t received = 0;
sf::IpAddress sender;
unsigned short port;
socket.receive(buffer, sizeof(buffer), received, sender, port);
std::cout << sender.toString() << " said: " << buffer << std::endl;

// ----- The server -----
// Create a socket and bind it to the port 55002
sf::UdpSocket socket;
socket.bind(55002);

// Receive a message from anyone
char buffer[1024];
std::size_t received = 0;
sf::IpAddress sender;
unsigned short port;
socket.receive(buffer, sizeof(buffer), received, sender, port);
std::cout << sender.toString() << " said: " << buffer << std::endl;

// Send an answer
std::string message = "Welcome " + sender.toString();
socket.send(message.c_str(), message.size() + 1, sender, port);
```

See also

[sf::Socket](#), [sf::TcpSocket](#), [sf::Packet](#)

Definition at line 45 of file [UdpSocket.hpp](#).

Member Enumeration Documentation

anonymous enum

Some special values used by sockets.

Enumerator

AnyPort

Special value that tells the system to pick any available port.

Definition at line 66 of file [Socket.hpp](#).

anonymous enum

Enumerator

MaxDatagramSize

The maximum number of bytes that can be sent in

Definition at line 52 of file [UdpSocket.hpp](#).

enum sf::Socket::Status

Status codes that may be returned by socket functions.

Enumerator

Done

	The socket has sent / received the data.
NotReady	The socket is not ready to send / receive data yet.
Partial	The socket sent a part of the data.
Disconnected	The TCP socket has been disconnected.
Error	An unexpected error happened.

Definition at line 53 of file [Socket.hpp](#).

enum sf::Socket::Type

Types of protocols that the socket can use.

Enumerator	
Tcp	TCP protocol.
Udp	UDP protocol.

Definition at line 114 of file [Socket.hpp](#).

Constructor & Destructor Documentation

sf::UdpSocket::UdpSocket()

Default constructor.

Member Function Documentation

```
Status sf::UdpSocket::bind ( unsigned short port,
                            const IpAddress & address = IpAddress()
                           )
```

Bind the socket to a specific port.

Binding the socket to a port is necessary for being able to receive data. Use the special value `Socket::AnyPort` to tell the system to automatically pick a port. Use `getLocalPort` to retrieve the chosen port.

Parameters

port Port to bind the socket to
address Address of the interface to bind to

Returns

Status code

See also

`unbind`, `getLocalPort`

```
void sf::Socket::close ( )
```

Close the socket gracefully.

This function can only be accessed by derived classes.

void sf::Socket::create ()

Create the internal representation of the socket.

This function can only be accessed by derived classes.

void sf::Socket::create (SocketHandle handle)

Create the internal representation of the socket from a socket handle.

This function can only be accessed by derived classes.

Parameters

handle OS-specific handle of the socket to wrap

SocketHandle sf::Socket::getHandle () const

Return the internal handle of the socket.

The returned handle may be invalid if the socket was not created yet.
This function can only be accessed by derived classes.

Returns

The internal (OS-specific) handle of the socket

unsigned short sf::UdpSocket::getLocalPort () const

Get the port to which the socket is bound locally.

If the socket is not bound to a port, this function returns 0.

Returns

Port to which the socket is bound

See also

[bind](#)

bool sf::Socket::isBlocking() const

Tell whether the socket is in blocking or non-blocking mode.

Returns

True if the socket is blocking, false otherwise

See also

[setBlocking](#)

```
Status sf::UdpSocket::receive ( void *           data,
                                std::size_t        size,
                                std::size_t &      received,
                                IpAddress &       remoteAddress,
                                unsigned short &   remotePort
                            )
```

Receive raw data from a remote peer.

In blocking mode, this function will wait until some bytes are actually received, which is large enough for the data that you intend to receive, if it is not returned and all the data will be lost.

Parameters

data	Pointer to the array to fill with the received bytes
size	Maximum number of bytes that can be received
received	This variable is filled with the actual number of bytes received.
remoteAddress	Address of the peer that sent the data
remotePort	Port of the peer that sent the data

Returns

Status code

See also

[send](#)

```
Status sf::UdpSocket::receive ( Packet & packet,
                                IpAddress & remoteAddress,
                                unsigned short & remotePort
                            )
```

Receive a formatted packet of data from a remote peer.

In blocking mode, this function will wait until the whole packet has been received.

Parameters

packet	Packet to fill with the received data
remoteAddress	Address of the peer that sent the data
remotePort	Port of the peer that sent the data

Returns

Status code

See also

[send](#)

```
Status sf::UdpSocket::send ( const void *           data,
                           std::size_t            size,
                           const IpAddress &    remoteAddress,
                           unsigned short         remotePort
                         )
```

Send raw data to a remote peer.

Make sure that `size` is not greater than `UdpSocket::MaxDatagramSize` and no data will be sent.

Parameters

<code>data</code>	Pointer to the sequence of bytes to send
<code>size</code>	Number of bytes to send
<code>remoteAddress</code>	Address of the receiver
<code>remotePort</code>	Port of the receiver to send the data to

Returns

Status code

See also

`receive`

```
Status sf::UdpSocket::send ( Packet &           packet,
                           const IpAddress & remoteAddress,
                           unsigned short     remotePort
                         )
```

Send a formatted packet of data to a remote peer.

Make sure that the packet size is not greater than `UdpSocket::MaxData`

will fail and no data will be sent.

Parameters

packet Packet to send

remoteAddress Address of the receiver

remotePort Port of the receiver to send the data to

Returns

Status code

See also

[receive](#)

void sf::Socket::setBlocking (bool blocking)

Set the blocking state of the socket.

In blocking mode, calls will not return until they have completed their task. In non-blocking mode won't return until some data was actually received. It always return immediately, using the return code to signal whether the operation succeeded or failed. By default, all sockets are blocking.

Parameters

blocking True to set the socket as blocking, false for non-blocking

See also

[isBlocking](#)

void sf::UdpSocket::unbind ()

Unbind the socket from the local port to which it is bound.

The port that the socket was previously bound to is immediately made available after this function is called. This means that a subsequent call to [bind\(\)](#) will fail if no other process has done so in the mean time. If the socket is not bound, this function has no effect.

See also

[bind](#)

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

- [UdpSocket.hpp](#)

SFML is licensed under the terms and conditions of the [zlib/png license](#).

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#).

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

[Classes](#) | [Functions](#)

System module

Base module of SFML, defining various utilities. More...

Classes

- | | |
|-------|--|
| class | sf::Clock
Utility class that measures the elapsed time. More... |
| class | sf::FileInputStream
Implementation of input stream based on a file. More... |
| class | sf::InputStream
Abstract class for custom file input streams. More... |
| class | sf::Lock
Automatic wrapper for locking and unlocking mutexes. More... |
| class | sf::MemoryInputStream
Implementation of input stream based on a memory chunk. More... |
| class | sf::Mutex
Blocks concurrent access to shared resources from multiple threads. More... |
| class | sf::NonCopyable
Utility class that makes any derived class non-copyable. More... |
| class | sf::String
Utility string class that automatically handles conversions between strings and C strings. More... |
| class | sf::Thread
Utility class to manipulate threads. More... |
| class | sf::ThreadLocal
Defines variables with thread-local storage. More... |
| class | sf::ThreadLocalPtr< T > |

Pointer to a thread-local variable. More...

class [sf::Time](#)

Represents a time value. More...

class [sf::Utf< N >](#)

Utility class providing generic functions for UTF conversions. More...

class [sf::Vector2< T >](#)

Utility template class for manipulating 2-dimensional vectors. More...

class [sf::Vector3< T >](#)

Utility template class for manipulating 3-dimensional vectors. More...

Functions

`ANativeActivity * sf::getNativeActivity ()`

Return a pointer to the Android native activity. [More...](#)

`void sf::sleep (Time duration)`

Make the current thread sleep for a given duration. [Mo](#)

`std::ostream & sf::err ()`

Standard stream used by SFML to output warnings and

Detailed Description

Base module of SFML, defining various utilities.

It provides vector classes, Unicode strings and conversion functions, three-

Function Documentation

sf::err()

Standard stream used by SFML to output warnings and errors.

By default, `sf::err()` outputs to the same location as `std::cerr`, (-> the console if there's one available).

It is a standard `std::ostream` instance, so it supports all the insertion (operator `<<`, manipulators, etc.).

`sf::err()` can be redirected to write to another output, independently c function provided by the `std::ostream` class.

Example:

```
// Redirect to a file
std::ofstream file("sfml-log.txt");
std::streambuf* previous = sf::err().rdbuf(file.rdbuf());

// Redirect to nothing
sf::err().rdbuf(NULL);

// Restore the original output
sf::err().rdbuf(previous);
```

Returns

Reference to `std::ostream` representing the SFML error stream

ANativeActivity* sf::getNativeActivity()

Return a pointer to the Android native activity.

You shouldn't have to use this function, unless you want to implement it, doesn't support, or to use a workaround for a known issue.

Returns

Pointer to Android native activity structure

Platform Limitation

This is only available on Android and to use it, you'll have to specify `SFML/System/NativeActivity.hpp` in your code.

`void sf::sleep (Time duration)`

Make the current thread sleep for a given duration.

`sf::sleep` is the best way to block a program or one of its threads, as it does not wake up until the specified duration has passed.

Parameters

duration Time to sleep

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [List of all members](#)

sf::Clock Class Reference

[System module](#)

Utility class that measures the elapsed time. [More...](#)

```
#include <Clock.hpp>
```

Public Member Functions

[Clock \(\)](#)

Default constructor. [More...](#)

[Time getElapsedTime \(\) const](#)

Get the elapsed time. [More...](#)

[Time restart \(\)](#)

Restart the clock. [More...](#)

Detailed Description

Utility class that measures the elapsed time.

`sf::Clock` is a lightweight class for measuring time.

It provides the most precise time that the underlying OS can achieve (nanoseconds). It also ensures monotonicity, which means that the returned value will never decrease even if the system time is changed.

Usage example:

```
sf::Clock clock;
...
Time time1 = clock.getElapsedTime();
...
Time time2 = clock.restart();
```

The `sf::Time` value returned by the clock can then be converted to a number of seconds, even microseconds.

See also

`sf::Time`

Definition at line 41 of file `Clock.hpp`.

Constructor & Destructor Documentation

sf::Clock::Clock()

Default constructor.

The clock starts automatically after being constructed.

Member Function Documentation

Time sf::Clock::getElapsedTime() const

Get the elapsed time.

This function returns the time elapsed since the last call to `restart()` (or `restart()` has not been called).

Returns

`Time` elapsed

Time sf::Clock::restart()

Restart the clock.

This function puts the time counter back to zero. It also returns the time started.

Returns

`Time` elapsed

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

- `Clock.hpp`
-

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [List of all members](#)

sf::FileInputStream Class Reference

[System module](#)

Implementation of input stream based on a file. [More...](#)

```
#include <FileInputStream.hpp>
```

Inheritance diagram for sf::FileInputStream:



Public Member Functions

`FileInputStream ()`

Default constructor. [More...](#)

`virtual ~FileInputStream ()`

Default destructor. [More...](#)

`bool open (const std::string &filename)`

Open the stream from a file path. [More...](#)

`virtual Int64 read (void *data, Int64 size)`

Read data from the stream. [More...](#)

`virtual Int64 seek (Int64 position)`

Change the current reading position. [More...](#)

`virtual Int64 tell ()`

Get the current reading position in the stream. [More...](#)

`virtual Int64 getSize ()`

Return the size of the stream. [More...](#)

Detailed Description

Implementation of input stream based on a file.

This class is a specialization of [InputStream](#) that reads from a file on disk.

It wraps a file in the common [InputStream](#) interface and therefore a functions that accept such a stream, with a file on disk as the data source.

In addition to the virtual functions inherited from [InputStream](#), [FileInputStream](#) opens the file to open.

SFML resource classes can usually be loaded directly from a filename, so you unless you create your own algorithms that operate on an [InputStream](#).

Usage example:

```
void process(InputStream& stream);
{
    FileInputStream stream;
    if (stream.open("some_file.dat"))
        process(stream);
```

[InputStream](#), [MemoryInputStream](#)

Definition at line 55 of file [FileInputStream.hpp](#).

Constructor & Destructor Documentation

sf::FileInputStream::FileInputStream ()

Default constructor.

virtual sf::FileInputStream::~FileInputStream ()

Default destructor.

Member Function Documentation

virtual Int64 sf::FileInputStream::getSize ()

Return the size of the stream.

Returns

The total number of bytes available in the stream, or -1 on error

Implements [sf::InputStream](#).

bool sf::FileInputStream::open (const std::string & filename)

Open the stream from a file path.

Parameters

filename Name of the file to open

Returns

True on success, false on error

virtual Int64 sf::FileInputStream::read (void * data, Int64 size)

Read data from the stream.

After reading, the stream's reading position must be advanced by the amount of bytes read.

Parameters

data Buffer where to copy the read data

size Desired number of bytes to read

Returns

The number of bytes actually read, or -1 on error

Implements [sf::InputStream](#).

virtual Int64 sf::FileInputStream::seek (Int64 position)

Change the current reading position.

Parameters

position The position to seek to, from the beginning

Returns

The position actually sought to, or -1 on error

Implements [sf::InputStream](#).

virtual Int64 sf::FileInputStream::tell ()

Get the current reading position in the stream.

Returns

The current position, or -1 on error.

Implements [sf::InputStream](#).

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

- [FileInputStream.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [List of all members](#)

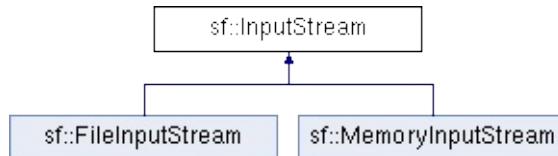
sf::InputStream Class Reference abstract

[System module](#)

Abstract class for custom file input streams. [More...](#)

```
#include <InputStream.hpp>
```

Inheritance diagram for sf::InputStream:



Public Member Functions

virtual ~InputStream ()
Virtual destructor. [More...](#)

virtual Int64 read (void *data, Int64 size)=0
Read data from the stream. [More...](#)

virtual Int64 seek (Int64 position)=0
Change the current reading position. [More...](#)

virtual Int64 tell ()=0
Get the current reading position in the stream. [More...](#)

virtual Int64 getSize ()=0
Return the size of the stream. [More...](#)

Detailed Description

Abstract class for custom file input streams.

This class allows users to define their own file input sources from which S

SFML resource classes like `sf::Texture` and `sf::SoundBuffer` provide load functions, which read data from conventional sources. However, if you have a source (over a network, embedded, encrypted, compressed, etc) you can use `sf::InputStream` and load SFML resources with their `loadFromStream` function.

Usage example:

```
// custom stream class that reads from inside a zip file
class ZipStream : public sf::InputStream
{
public:
    ZipStream(std::string archive);

    bool open(std::string filename);

    Int64 read(void* data, Int64 size);

    Int64 seek(Int64 position);

    Int64 tell();

    Int64 getSize();

private:
    ...
};

// now you can load textures...
sf::Texture texture;
ZipStream stream("resources.zip");
stream.open("images/img.png");
texture.loadFromStream(stream);

// musics...
sf::Music music;
```

```
ZipStream stream("resources.zip");
stream.open("musics/msc.ogg");
music.openFromStream(stream);

// etc.
```

Definition at line 41 of file `InputStream.hpp`.

Constructor & Destructor Documentation

virtual sf::InputStream::~InputStream()

Virtual destructor.

Definition at line [49](#) of file [InputStream.hpp](#).

Member Function Documentation

virtual Int64 sf::InputStream::getSize ()

Return the size of the stream.

Returns

The total number of bytes available in the stream, or -1 on error

Implemented in [sf::FileInputStream](#), and [sf::MemoryInputStream](#).

virtual Int64 sf::InputStream::read (void * data, Int64 size)

Read data from the stream.

After reading, the stream's reading position must be advanced by the amount of data read.

Parameters

data Buffer where to copy the read data
size Desired number of bytes to read

Returns

The number of bytes actually read, or -1 on error

Implemented in [sf::FileInputStream](#), and [sf::MemoryInputStream](#).

virtual Int64 sf::InputStream::seek (Int64 position)

Change the current reading position.

Parameters

position The position to seek to, from the beginning

Returns

The position actually sought to, or -1 on error

Implemented in [sf::FileInputStream](#), and [sf::MemoryInputStream](#).

virtual Int64 sf::InputStream::tell ()

Get the current reading position in the stream.

Returns

The current position, or -1 on error.

Implemented in [sf::FileInputStream](#), and [sf::MemoryInputStream](#).

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

- [InputStream.hpp](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [List of all members](#)

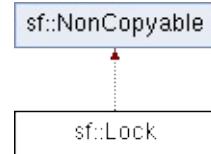
sf::Lock Class Reference

[System module](#)

Automatic wrapper for locking and unlocking mutexes. More...

```
#include <Lock.hpp>
```

Inheritance diagram for sf::Lock:



Public Member Functions

`Lock (Mutex &mutex)`

Construct the lock with a target mutex. More...

`~Lock ()`

Destructor. More...

Detailed Description

Automatic wrapper for locking and unlocking mutexes.

`sf::Lock` is a RAII wrapper for `sf::Mutex`.

By unlocking it in its destructor, it ensures that the mutex will always be (most likely a function) ends. This is even more important when an exception can interrupt the execution flow of the function.

For maximum robustness, `sf::Lock` should always be used to lock/unlock a mutex.

Usage example:

```
sf::Mutex mutex;

void function()
{
    sf::Lock lock(mutex); // mutex is now locked

    functionThatMayThrowAnException(); // mutex is unlocked if this function throws

    if (someCondition)
        return; // mutex is unlocked

} // mutex is unlocked
```

Because the mutex is not explicitly unlocked in the code, it may remain locked after the function returns. If the region of the code that needs to be protected by the mutex is not the entire function, it is better to create a smaller, inner scope so that the lock is limited to this part of the code.

```
sf::Mutex mutex;

void function()
{
    {
        sf::Lock lock(mutex);
        codeThatRequiresProtection();
    }
}
```

```
    } // mutex is unlocked here  
    codeThatDoesntCareAboutTheMutex();  
}
```

Having a mutex locked longer than required is a bad practice which can forget that when a mutex is locked, other threads may be waiting doing nc

See also

[sf::Mutex](#)

Definition at line 43 of file [Lock.hpp](#).

Constructor & Destructor Documentation

sf::Lock::Lock (Mutex & mutex)

Construct the lock with a target mutex.

The mutex passed to `sf::Lock` is automatically locked.

Parameters

mutex Mutex to lock

sf::Lock::~Lock ()

Destructor.

The destructor of `sf::Lock` automatically unlocks its mutex.

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

- `Lock.hpp`
-

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [List of all members](#)

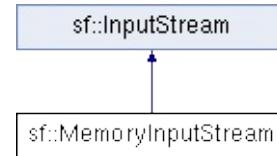
sf::MemoryInputStream Class Reference

[System module](#)

Implementation of input stream based on a memory chunk. [More...](#)

```
#include <MemoryInputStream.hpp>
```

Inheritance diagram for sf::MemoryInputStream:



Public Member Functions

[MemoryInputStream \(\)](#)
Default constructor. [More...](#)

`void open (const void *data, std::size_t sizeInBytes)`
Open the stream from its data. [More...](#)

`virtual Int64 read (void *data, Int64 size)`
Read data from the stream. [More...](#)

`virtual Int64 seek (Int64 position)`
Change the current reading position. [More...](#)

`virtual Int64 tell ()`
Get the current reading position in the stream. [More...](#)

`virtual Int64 getSize ()`
Return the size of the stream. [More...](#)

Detailed Description

Implementation of input stream based on a memory chunk.

This class is a specialization of [InputStream](#) that reads from data in memory.

It wraps a memory chunk in the common [InputStream](#) interface and therefore provides all the same functions that accept such a stream, with content already loaded in memory.

In addition to the virtual functions inherited from [InputStream](#), [MemoryInputStream](#) also provides two additional functions to specify the pointer and size of the data in memory.

SFML resource classes can usually be loaded directly from memory, so this class is mainly useful unless you create your own algorithms that operate on an [InputStream](#).

Usage example:

```
void process(InputStream& stream);
MemoryInputStream stream;
stream.open(thePtr, theSize);
process(stream);
```

[InputStream](#), [FileInputStream](#)

Definition at line 43 of file [MemoryInputStream.hpp](#).

Constructor & Destructor Documentation

sf::MemoryInputStream::MemoryInputStream()

Default constructor.

Member Function Documentation

virtual Int64 sf::MemoryInputStream::getSize ()

Return the size of the stream.

Returns

The total number of bytes available in the stream, or -1 on error

Implements [sf::InputStream](#).

void sf::MemoryInputStream::open (const void * data, std::size_t sizeInBytes)

Open the stream from its data.

Parameters

data Pointer to the data in memory

sizeInBytes Size of the data, in bytes

virtual Int64 sf::MemoryInputStream::read (void * data, Int64 size)

Read data from the stream.

After reading, the stream's reading position must be advanced by the amount of bytes read.

Parameters

data Buffer where to copy the read data

size Desired number of bytes to read

Returns

The number of bytes actually read, or -1 on error

Implements [sf::InputStream](#).

virtual Int64 sf::MemoryInputStream::seek (Int64 position)

Change the current reading position.

Parameters

position The position to seek to, from the beginning

Returns

The position actually sought to, or -1 on error

Implements [sf::InputStream](#).

virtual Int64 sf::MemoryInputStream::tell ()

Get the current reading position in the stream.

Returns

The current position, or -1 on error.

Implements [sf::InputStream](#).

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

- [MemoryInputStream.hpp](#)
-

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [List of all members](#)

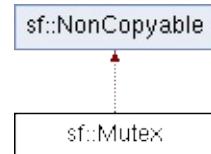
sf::Mutex Class Reference

[System module](#)

Blocks concurrent access to shared resources from multiple threads. More...

```
#include <Mutex.hpp>
```

Inheritance diagram for sf::Mutex:



Public Member Functions

`Mutex ()`

Default constructor. [More...](#)

`~Mutex ()`

Destructor. [More...](#)

`void lock ()`

Lock the mutex. [More...](#)

`void unlock ()`

Unlock the mutex. [More...](#)

Detailed Description

Blocks concurrent access to shared resources from multiple threads.

Mutex stands for "MUTual EXclusion".

A mutex is a synchronization object, used when multiple threads are involved.

When you want to protect a part of the code from being accessed simultaneously by multiple threads, you typically use a mutex. When a thread is locked by a mutex, any other thread trying to lock it will have to wait until the mutex is released by the thread that locked it. This way, you can safely access a critical region of your code.

Usage example:

```
Database database; // this is a critical resource that needs some protection
sf::Mutex mutex;

void thread1()
{
    mutex.lock(); // this call will block the thread if the mutex is already
    database.write(...);
    mutex.unlock(); // if thread2 was waiting, it will now be unblocked
}

void thread2()
{
    mutex.lock(); // this call will block the thread if the mutex is already
    database.write(...);
    mutex.unlock(); // if thread1 was waiting, it will now be unblocked
}
```

Be very careful with mutexes. A bad usage can lead to bad problems, like threads waiting for each other and the application is globally stuck).

To make the usage of mutexes more robust, particularly in environments where threads can be interrupted, you should use the helper class **sf::Lock** to lock/unlock mutexes.

SFML mutexes are recursive, which means that you can lock a mutex `r` without creating a deadlock. In this case, the first call to `lock()` behaves have no effect. However, you must call `unlock()` exactly as many times as mutex won't be released.

See also

`sf::Lock`

Definition at line 47 of file `Mutex.hpp`.

Constructor & Destructor Documentation

sf::Mutex::Mutex()

Default constructor.

sf::Mutex::~Mutex()

Destructor.

Member Function Documentation

void sf::Mutex::lock()

Lock the mutex.

If the mutex is already locked in another thread, this call will block until released.

See also

[unlock](#)

void sf::Mutex::unlock()

Unlock the mutex.

See also

[lock](#)

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

- [Mutex.hpp](#)
-

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Protected Member Functions](#) | [List of all members](#)

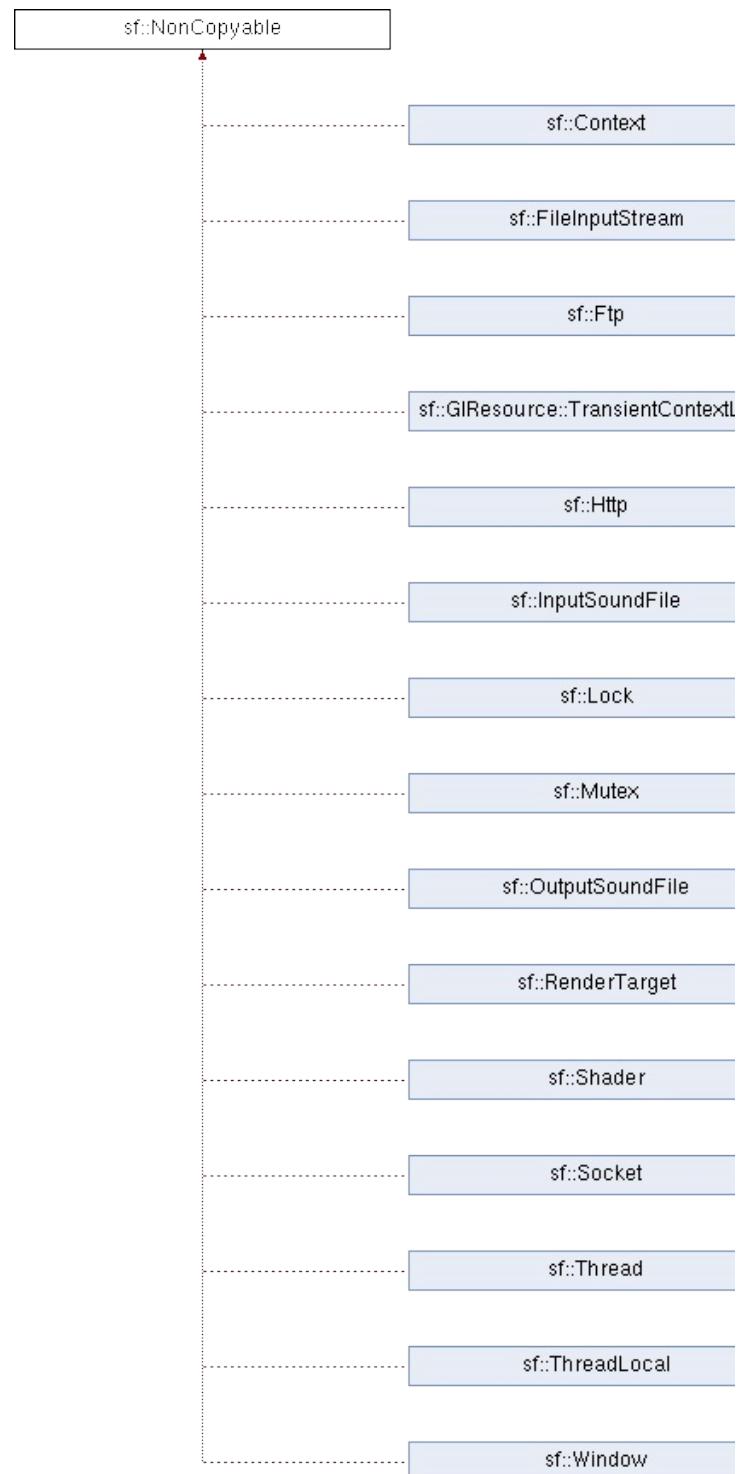
sf::NonCopyable Class Reference

[System module](#)

Utility class that makes any derived class non-copyable. More...

```
#include <NonCopyable.hpp>
```

Inheritance diagram for sf::NonCopyable:



Protected Member Functions

[NonCopyable \(\)](#)

Default constructor. More...

Detailed Description

Utility class that makes any derived class non-copyable.

This class makes its instances non-copyable, by explicitly disabling its copy operator.

To create a non-copyable class, simply inherit from `sf::NonCopyable`.

The type of inheritance (public or private) doesn't matter, the copy constructor is declared private in `sf::NonCopyable` so they will end up being inaccessible. There is also a shorter syntax for inheriting from it (see below).

Usage example:

```
class MyNonCopyableClass : sf::NonCopyable
{
    ...
};
```

Deciding whether the instances of a class can be copied or not is a very important decision. It is strongly encouraged to think about it before writing a class, and to use `sf::NonCopyable` to prevent many potential future errors when using it. This is also a very important decision for the class.

Definition at line 41 of file `NonCopyable.hpp`.

Constructor & Destructor Documentation

sf::NonCopyable::NonCopyable()

Default constructor.

Because this class has a copy constructor, the compiler will not automatically generate a default constructor. That's why we must define it explicitly.

Definition at line 53 of file [NonCopyable.hpp](#).

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

- [NonCopyable.hpp](#)

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#).

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Types](#) | [Public Member Functions](#) | [Static Public Member Functions](#) | [Static Public Attributes](#) | [Friends](#) | [Related Functions](#)

sf::String Class Reference

[System module](#)

Utility string class that automatically handles conversions between types ↴

```
#include <String.hpp>
```

Public Types

```
typedef std::basic_string  
< Uint32 >::iterator
```

Iterator

Iterator type. [More...](#)

```
typedef std::basic_string  
< Uint32 >::const_iterator
```

ConstIterator

Read-only iterator type. [More...](#)

Public Member Functions

[String \(\)](#)

Default constructor. [More...](#)

[String \(char ansiChar, const std::locale &loc\)](#)

Construct from a single ANSI character and a locale.

[String \(wchar_t wideChar\)](#)

Construct from single wide character. [More...](#)

[String \(Uint32 utf32Char\)](#)

Construct from single UTF-32 character. [More...](#)

[String \(const char *ansiString, const std::locale &loc\)](#)

Construct from a null-terminated C-style ANSI string and a locale.

[String \(const std::string &ansiString, const std::locale &loc\)](#)

Construct from an ANSI string and a locale.

[String \(const wchar_t *wideString\)](#)

Construct from null-terminated C-style wide string.

[String \(const std::wstring &wideString\)](#)

Construct from a wide string. [More...](#)

[String \(const Uint32 *utf32String\)](#)

Construct from a null-terminated C-style UTF-32 string.

[String \(const std::basic_string< Uint32 > &utf32String\)](#)

Construct from an UTF-32 string. [More...](#)

[String \(const String ©\)](#)

Copy constructor. More...

`operator std::string () const`
Implicit conversion operator to std::string (AN)

`operator std::wstring () const`
Implicit conversion operator to std::wstring (v)

`std::string toAnsiString (const std::locale &locale=std::locale()) const`
Convert the Unicode string to an ANSI string

`std::wstring toWideString () const`
Convert the Unicode string to a wide string. I

`std::basic_string< Uint8 > toUtf8 () const`
Convert the Unicode string to a UTF-8 string

`std::basic_string< Uint16 > toUtf16 () const`
Convert the Unicode string to a UTF-16 strin

`std::basic_string< Uint32 > toUtf32 () const`
Convert the Unicode string to a UTF-32 strin

`String & operator= (const String &right)`
Overload of assignment operator. More...

`String & operator+= (const String &right)`
Overload of += operator to append an UTF-3

`Uint32 operator[] (std::size_t index) const`
Overload of [] operator to access a character

`Uint32 & operator[] (std::size_t index)`
Overload of [] operator to access a character

void `clear ()`
Clear the string. [More...](#)

`std::size_t getSize () const`
Get the size of the string. [More...](#)

bool `isEmpty () const`
Check whether the string is empty or not. [More...](#)

void `erase (std::size_t position, std::size_t count=1)`
Erase one or more characters from the string.

void `insert (std::size_t position, const String &str)`
Insert one or more characters into the string.

`std::size_t find (const String &str, std::size_t start=0) const`
Find a sequence of one or more characters in the string.

void `replace (std::size_t position, std::size_t length, const String &str)`
Replace a substring with another string. [More...](#)

void `replace (const String &searchFor, const String &replaceWith)`
Replace all occurrences of a substring with another string.

`String substring (std::size_t position, std::size_t length)`
Return a part of the string. [More...](#)

`const Uint32 * getData () const`
Get a pointer to the C-style array of characters.

`Iterator begin ()`
Return an iterator to the beginning of the string.

`ConstIterator begin () const`
Return an iterator to the beginning of the string.

`Iterator end ()`

Return an iterator to the end of the string. [More...](#)

`ConstIterator end () const`

Return an iterator to the end of the string. [More...](#)

Static Public Member Functions

```
template<typename T >
static String fromUtf8 (T begin, T end)
    Create a new sf::String from a UTF-8 encoded string. More
```

```
template<typename T >
static String fromUtf16 (T begin, T end)
    Create a new sf::String from a UTF-16 encoded string. Mor
```

```
template<typename T >
static String fromUtf32 (T begin, T end)
    Create a new sf::String from a UTF-32 encoded string. Mor
```

Static Public Attributes

static const std::size_t `InvalidPos`

Represents an invalid position in the string. More

Friends

```
bool operator== (const String &left, const String &right)
```

```
bool operator< (const String &left, const String &right)
```

Related Functions

(Note that these are not member functions.)

`bool operator==(const String &left, const String &right)`
Overload of == operator to compare two UTF-32 strings. [More...](#)

`bool operator!=(const String &left, const String &right)`
Overload of != operator to compare two UTF-32 strings. [More...](#)

`bool operator<(const String &left, const String &right)`
Overload of < operator to compare two UTF-32 strings. [More...](#)

`bool operator>(const String &left, const String &right)`
Overload of > operator to compare two UTF-32 strings. [More...](#)

`bool operator<=(const String &left, const String &right)`
Overload of <= operator to compare two UTF-32 strings. [More...](#)

`bool operator>=(const String &left, const String &right)`
Overload of >= operator to compare two UTF-32 strings. [More...](#)

`String operator+(const String &left, const String &right)`
Overload of binary + operator to concatenate two strings. [More...](#)

Detailed Description

Utility string class that automatically handles conversions between types and

`sf::String` is a utility string class defined mainly for convenience.

It is a Unicode string (implemented using UTF-32), thus it can store any Chinese, Arabic, Hebrew, etc.).

It automatically handles conversions from/to ANSI and wide strings, so string classes and still be compatible with functions taking a `sf::String`.

```
sf::String s;

std::string s1 = s; // automatically converted to ANSI string
std::wstring s2 = s; // automatically converted to wide string
s = "hello"; // automatically converted from ANSI string
s = L"hello"; // automatically converted from wide string
s += 'a'; // automatically converted from ANSI string
s += L'a'; // automatically converted from wide string
```

Conversions involving ANSI strings use the default user locale. However, locale if necessary:

```
std::locale locale;
sf::String s;
...
std::string s1 = s.toAnsiString(locale);
s = sf::String("hello", locale);
```

`sf::String` defines the most important functions of the standard `std::string` like iterating, appending, comparing, etc. However it is a simple class provided by SFML, so you may have to consider using a more optimized class if your program requires it. The automatic conversion functions will then take care of converting your strings to the appropriate type if they require it.

Please note that SFML also defines a low-level, generic interface for `I` classes.

Definition at line 45 of file `String.hpp`.

Member Typedef Documentation

typedef std::basic_string<Uint32>::const_iterator sf::String::ConstIt

Read-only iterator type.

Definition at line 53 of file [String.hpp](#).

typedef std::basic_string<Uint32>::iterator sf::String::Iterator

Iterator type.

Definition at line 52 of file [String.hpp](#).

Constructor & Destructor Documentation

`sf::String::String()`

Default constructor.

This constructor creates an empty string.

```
sf::String::String( char               ansiChar,  
                    const std::locale & locale = std::locale()  
                )
```

Construct from a single ANSI character and a locale.

The source character is converted to UTF-32 according to the given locale.

Parameters

ansiChar ANSI character to convert
locale Locale to use for conversion

`sf::String::String(wchar_t wideChar)`

Construct from single wide character.

Parameters

wideChar Wide character to convert

sf::String::String (Uint32 utf32Char)

Construct from single UTF-32 character.

Parameters

utf32Char UTF-32 character to convert

sf::String::String (const char * ansiString, const std::locale & locale = std::locale())

Construct from a null-terminated C-style ANSI string and a locale.

The source string is converted to UTF-32 according to the given locale.

Parameters

ansiString ANSI string to convert
locale Locale to use for conversion

sf::String::String (const std::string & ansiString, const std::locale & locale = std::locale())

Construct from an ANSI string and a locale.

The source string is converted to UTF-32 according to the given locale.

Parameters

ansiString ANSI string to convert

locale Locale to use for conversion

sf::String::String (const wchar_t * wideString)

Construct from null-terminated C-style wide string.

Parameters

wideString Wide string to convert

sf::String::String (const std::wstring & wideString)

Construct from a wide string.

Parameters

wideString Wide string to convert

sf::String::String (const Uint32 * utf32String)

Construct from a null-terminated C-style UTF-32 string.

Parameters

utf32String UTF-32 string to assign

sf::String::String (const std::basic_string< Uint32 > & utf32String)

Construct from an UTF-32 string.

Parameters

utf32String UTF-32 string to assign

sf::String::String (const String & **copy)**

Copy constructor.

Parameters

copy Instance to copy

Member Function Documentation

Iterator `sf::String::begin()`

Return an iterator to the beginning of the string.

Returns

Read-write iterator to the beginning of the string characters

See also

`end`

ConstIterator `sf::String::begin() const`

Return an iterator to the beginning of the string.

Returns

Read-only iterator to the beginning of the string characters

See also

`end`

void `sf::String::clear()`

Clear the string.

This function removes all the characters from the string.

See also

[isEmpty](#), [erase](#)

Iterator `sf::String::end()`

Return an iterator to the end of the string.

The end iterator refers to 1 position past the last character; thus it represents an invalid position and should never be accessed.

Returns

Read-write iterator to the end of the string characters

See also

[begin](#)

ConstIterator `sf::String::end() const`

Return an iterator to the end of the string.

The end iterator refers to 1 position past the last character; thus it represents an invalid position and should never be accessed.

Returns

Read-only iterator to the end of the string characters

See also

[begin](#)

```
void sf::String::erase ( std::size_t position,  
                        std::size_t count = 1  
)
```

Erase one or more characters from the string.

This function removes a sequence of *count* characters starting from *posi*

Parameters

position Position of the first character to erase
count Number of characters to erase

```
std::size_t sf::String::find ( const String & str,  
                            std::size_t start = 0  
) const
```

Find a sequence of one or more characters in the string.

This function searches for the characters of *str* in the string, starting from

Parameters

str Characters to find
start Where to begin searching

Returns

Position of *str* in the string, or *String::InvalidPos* if not found

```
template<typename T >  
static String sf::String::fromUtf16 ( T begin,
```

T end

)

Create a new sf::String from a UTF-16 encoded string.

Parameters

- begin** Forward iterator to the beginning of the UTF-16 sequence
- end** Forward iterator to the end of the UTF-16 sequence

Returns

A sf::String containing the source string

See also

[fromUtf8](#), [fromUtf32](#)

```
template<typename T >
```

```
static String sf::String::fromUtf32 (T begin,
                                     T end
                                     )
```

Create a new sf::String from a UTF-32 encoded string.

This function is provided for consistency, it is equivalent to using the sf::Uint32* or a std::basic_string<sf::Uint32>.

Parameters

- begin** Forward iterator to the beginning of the UTF-32 sequence
- end** Forward iterator to the end of the UTF-32 sequence

Returns

A sf::String containing the source string

See also

[fromUtf8](#), [fromUtf16](#)

```
template<typename T >
static String sf::String::fromUtf8 ( T begin,
                                T end
)
```

Create a new [sf::String](#) from a UTF-8 encoded string.

Parameters

begin Forward iterator to the beginning of the UTF-8 sequence
end Forward iterator to the end of the UTF-8 sequence

Returns

A [sf::String](#) containing the source string

See also

[fromUtf16](#), [fromUtf32](#)

```
const Uint32* sf::String::getData ( ) const
```

Get a pointer to the C-style array of characters.

This functions provides a read-only access to a null-terminated C-style returned pointer is temporary and is meant only for immediate use, thus it

Returns

Read-only pointer to the array of characters

`std::size_t sf::String::getSize() const`

Get the size of the string.

Returns

Number of characters in the string

See also

[isEmpty](#)

```
void sf::String::insert( std::size_t position,  
                        const String & str  
                      )
```

Insert one or more characters into the string.

This function inserts the characters of *str* into the string, starting from *pos*.

Parameters

position Position of insertion

str Characters to insert

`bool sf::String::isEmpty() const`

Check whether the string is empty or not.

Returns

True if the string is empty (i.e. contains no character)

See also

[clear](#), [getSize](#)

sf::String::operator std::string() const

Implicit conversion operator to std::string (ANSI string)

The current global locale is used for conversion. If you want to use a different locale, use [toAnsiString](#). Characters that do not fit in the target encoding are discarded from the string. This operator is defined for convenience, and is equivalent to calling [toAnsiString\(\)](#).

Returns

Converted ANSI string

See also

[toAnsiString](#), [operator std::wstring](#)

sf::String::operator std::wstring() const

Implicit conversion operator to std::wstring (wide string)

Characters that do not fit in the target encoding are discarded from the string. This operator is defined for convenience, and is equivalent to calling [toWideString\(\)](#).

Returns

Converted wide string

See also

[toWideString](#), [operator std::string](#)

String& sf::String::operator+=(const String & right)

Overload of += operator to append an UTF-32 string.

Parameters

right String to append

Returns

Reference to self

String& sf::String::operator= (const String & right)

Overload of assignment operator.

Parameters

right Instance to assign

Returns

Reference to self

Uint32 sf::String::operator[] (std::size_t index) const

Overload of [] operator to access a character by its position.

This function provides read-only access to characters. Note: the behavior depends on the range.

Parameters

index Index of the character to get

Returns

Character at position *index*

Uint32& sf::String::operator[] (std::size_t index)

Overload of [] operator to access a character by its position.

This function provides read and write access to characters. Note: the behavior of range.

Parameters

index Index of the character to get

Returns

Reference to the character at position *index*

```
void sf::String::replace ( std::size_t position,  
                           std::size_t length,  
                           const String & replaceWith  
                         )
```

Replace a substring with another string.

This function replaces the substring that starts at index *position* and has length *length* with string *replaceWith*.

Parameters

position Index of the first character to be replaced

length Number of characters to replace. You can pass Invalid until the end of the string.

replaceWith String that replaces the given substring.

```
void sf::String::replace ( const String & searchFor,  
                         const String & replaceWith  
                     )
```

Replace all occurrences of a substring with a replacement string.

This function replaces all occurrences of *searchFor* in this string with the

Parameters

searchFor The value being searched for

replaceWith The value that replaces found *searchFor* values

```
String sf::String::substring ( std::size_t position,  
                             std::size_t length = InvalidPos  
                           ) const
```

Return a part of the string.

This function returns the substring that starts at index *position* and spans

Parameters

position Index of the first character

length Number of characters to include in the substring (if the string ends before that many characters as possible are included). **InvalidPos** can be used to indicate the end of the string.

Returns

String object containing a substring of this object

```
std::string sf::String::toAnsiString ( const std::locale & locale = std::
```

Convert the Unicode string to an ANSI string.

The UTF-32 string is converted to an ANSI string in the encoding defined by the locale. Characters that do not fit in the target encoding are discarded from the returned string.

Parameters

locale Locale to use for conversion

Returns

Converted ANSI string

See also

[toWideString](#), operator std::string

`std::basic_string<UInt16> sf::String::toUtf16() const`

Convert the Unicode string to a UTF-16 string.

Returns

Converted UTF-16 string

See also

[toUtf8](#), [toUtf32](#)

`std::basic_string<UInt32> sf::String::toUtf32() const`

Convert the Unicode string to a UTF-32 string.

This function doesn't perform any conversion, since the string is already

Returns

Converted UTF-32 string

See also

[toUtf8](#), [toUtf16](#)

std::basic_string<UInt8> sf::String::toUtf8 () const

Convert the Unicode string to a UTF-8 string.

Returns

Converted UTF-8 string

See also

[toUtf16](#), [toUtf32](#)

std::wstring sf::String::toWideString () const

Convert the Unicode string to a wide string.

Characters that do not fit in the target encoding are discarded from the resulting string.

Returns

Converted wide string

See also

[toAnsiString](#), operator std::wstring

Friends And Related Function Documentation

```
bool operator!= ( const String & left,  
                  const String & right  
                )
```

Overload of != operator to compare two UTF-32 strings.

Parameters

left Left operand (a string)
right Right operand (a string)

Returns

True if both strings are different

```
String operator+ ( const String & left,  
                   const String & right  
                 )
```

Overload of binary + operator to concatenate two strings.

Parameters

left Left operand (a string)
right Right operand (a string)

Returns

Concatenated string

```
bool operator< ( const String & left,  
                  const String & right  
                )
```

Overload of < operator to compare two UTF-32 strings.

Parameters

left Left operand (a string)
right Right operand (a string)

Returns

True if *left* is lexicographically before *right*

```
bool operator<= ( const String & left,  
                   const String & right  
                 )
```

Overload of <= operator to compare two UTF-32 strings.

Parameters

left Left operand (a string)
right Right operand (a string)

Returns

True if *left* is lexicographically before or equivalent to *right*

```
bool operator== ( const String & left,  
                   const String & right  
                 )
```

Overload of == operator to compare two UTF-32 strings.

Parameters

left Left operand (a string)
right Right operand (a string)

Returns

True if both strings are equal

```
bool operator> ( const String & left,  
                  const String & right  
                )
```

Overload of > operator to compare two UTF-32 strings.

Parameters

left Left operand (a string)
right Right operand (a string)

Returns

True if *left* is lexicographically after *right*

```
bool operator>= ( const String & left,  
                   const String & right  
                 )
```

Overload of >= operator to compare two UTF-32 strings.

Parameters

left Left operand (a string)

right Right operand (a string)

Returns

True if *left* is lexicographically after or equivalent to *right*

Member Data Documentation

const std::size_t sf::String::InvalidPos

Represents an invalid position in the string.

Definition at line 58 of file [String.hpp](#).

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

- [String.hpp](#)
-

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Member Functions](#) | [List of all members](#)

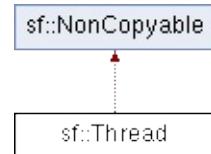
sf::Thread Class Reference

[System module](#)

Utility class to manipulate threads. More...

```
#include <Thread.hpp>
```

Inheritance diagram for sf::Thread:



Public Member Functions

template<typename F >

Thread (F function)

Construct the thread from a functor with no argument. [More...](#)

template<typename F , typename A >

Thread (F function, A argument)

Construct the thread from a functor with an argument. [More...](#)

template<typename C >

Thread (void(C::*function)(), C *object)

Construct the thread from a member function and an object. [More..](#)

~Thread ()

Destructor. [More...](#)

void launch ()

Run the thread. [More...](#)

void wait ()

Wait until the thread finishes. [More...](#)

void terminate ()

Terminate the thread. [More...](#)

Detailed Description

Utility class to manipulate threads.

Threads provide a way to run multiple parts of the code in parallel.

When you launch a new thread, the execution is split and both the new thi

To use a `sf::Thread`, you construct it directly with the function to execute. `sf::Thread` has multiple template constructors, which means that you can

- non-member functions with no argument
- non-member functions with one argument of any type
- functors with no argument (this one is particularly useful for compatibility)
- functors with one argument of any type
- member functions from any class with no argument

The function argument, if any, is copied in the `sf::Thread` instance, corresponding constructor is used). Class instances, however, are passed by pointer, so make sure that the object won't be destroyed while the thread is still using it.

The thread ends when its function is terminated. If the owner `sf::Thread` is destroyed before the thread is finished, the destructor will wait (see `wait()`)

Usage examples:

```
// example 1: non member function with one argument
void threadFunc(int argument)
{
    ...
}
sf::Thread thread(&threadFunc, 5);
```

```

thread.launch(); // start the thread (internally calls threadFunc(5))

// example 2: member function

class Task
{
public:
    void run()
    {
        ...
    }
};

Task task;
sf::Thread thread(&Task::run, &task);
thread.launch(); // start the thread (internally calls task.run())

// example 3: functor

struct Task
{
    void operator()()
    {
        ...
    }
};

sf::Thread thread(Task());
thread.launch(); // start the thread (internally calls operator() on th

```

Creating parallel threads of execution can be dangerous: all threads in the same memory space, which means that you may end up accessing the same memory at the same time. To prevent this kind of situations, you can use mutexes.

See also

[sf::Mutex](#)

Definition at line 48 of file [Thread.hpp](#).

Constructor & Destructor Documentation

```
template<typename F >
```

sf::Thread::Thread (F function)

Construct the thread from a functor with no argument.

This constructor works for function objects, as well as free functions.

Use this constructor for this kind of function:

```
void function();  
// --- or ----  
struct Functor  
{  
    void operator()();  
};
```

Note: this does *not* run the thread, use `launch()`.

Parameters

function Functor or free function to use as the entry point of the thread

```
template<typename F , typename A >
```

sf::Thread::Thread (F function, A argument)

Construct the thread from a functor with an argument.

This constructor works for function objects, as well as free functions. It the argument can have any type (int, std::string, void*, Toto, ...).

Use this constructor for this kind of function:

```
void function(int arg);
// --- or ----
struct Functor
{
    void operator()(std::string arg);
};
```

Note: this does *not* run the thread, use `launch()`.

Parameters

function Functor or free function to use as the entry point of the thread
argument argument to forward to the function

```
template<typename C >
sf::Thread::Thread ( void(C::*)( ) function,
                    C *          object
                  )
```

Construct the thread from a member function and an object.

This constructor is a template, which means that you can use it with any this kind of function:

```
class MyClass
{
public:
    void function();
};
```

Note: this does *not* run the thread, use `launch()`.

Parameters

- function** Entry point of the thread
- object** Pointer to the object to use

`sf::Thread::~Thread()`

Destructor.

This destructor calls `wait()`, so that the internal thread cannot survive destroyed.

Member Function Documentation

void sf::Thread::launch()

Run the thread.

This function starts the entry point passed to the thread's constructor, and if the function returns, the thread's function is running in parallel to the calling one.

void sf::Thread::terminate()

Terminate the thread.

This function immediately stops the thread, without waiting for its function to return. Using this function is not safe, and can lead to local variables not being properly destroyed in multi-threaded systems. You should rather try to make the thread function terminate by itself.

void sf::Thread::wait()

Wait until the thread finishes.

This function will block the execution until the thread's function ends. If the function never ends, the calling thread will block forever. If this function is called while another thread is already waiting, it will do nothing.

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

- [Thread.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

Classes

[Files](#)

[Class List](#)

[Class Index](#)

[Class Hierarchy](#)

[Class Members](#)

[Public Member Functions](#) | [List of all members](#)

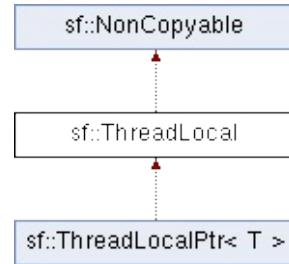
sf::ThreadLocal Class Reference

[System module](#)

Defines variables with thread-local storage. [More...](#)

```
#include <ThreadLocal.hpp>
```

Inheritance diagram for sf::ThreadLocal:



Public Member Functions

`ThreadLocal (void *value=NULL)`

Default constructor. [More...](#)

`~ThreadLocal ()`

Destructor. [More...](#)

`void setValue (void *value)`

Set the thread-specific value of the variable. [More...](#)

`void * getValue () const`

Retrieve the thread-specific value of the variable. [More...](#)

Detailed Description

Defines variables with thread-local storage.

This class manipulates void* parameters and thus is not appropriate for st

You should rather use the `sf::ThreadLocalPtr` template class.

Definition at line 47 of file `ThreadLocal.hpp`.

Constructor & Destructor Documentation

sf::ThreadLocal::ThreadLocal (void * value = NULL)

Default constructor.

Parameters

value Optional value to initialize the variable

sf::ThreadLocal::~ThreadLocal ()

Destructor.

Member Function Documentation

void* sf::ThreadLocal::getValue() const

Retrieve the thread-specific value of the variable.

Returns

Value of the variable for the current thread

void sf::ThreadLocal::setValue(void * value)

Set the thread-specific value of the variable.

Parameters

value Value of the variable for the current thread

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

- ThreadLocal.hpp

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Private Member Functions](#) | [List of all members](#)

`sf::ThreadLocalPtr< T >` Class Template Reference

[System module](#)

Pointer to a thread-local variable. [More...](#)

```
#include <ThreadLocalPtr.hpp>
```

Inheritance diagram for `sf::ThreadLocalPtr< T >`:



Public Member Functions

`ThreadLocalPtr (T *value=NULL)`
Default constructor. [More...](#)

`T & operator* () const`
Overload of unary operator *. [More...](#)

`T * operator-> () const`
Overload of operator ->. [More...](#)

`operator T * () const`
Conversion operator to implicitly convert the poi
[More...](#)

`ThreadLocalPtr< T > & operator= (T *value)`
Assignment operator for a raw pointer parameter

`ThreadLocalPtr< T > & operator= (const ThreadLocalPtr< T > &right)`
Assignment operator for a `ThreadLocalPtr` parame

Private Member Functions

`void setValue (void *value)`

Set the thread-specific value of the variable. [More...](#)

`void * getValue () const`

Retrieve the thread-specific value of the variable. [More...](#)

Detailed Description

```
template<typename T>
class sf::ThreadLocalPtr< T >
```

Pointer to a thread-local variable.

`sf::ThreadLocalPtr` is a type-safe wrapper for storing pointers to thread-loc

A thread-local variable holds a different value for each different thread, shared.

Its usage is completely transparent, so that it is similar to manipulating smart pointer).

Usage example:

```
MyClass object1;
MyClass object2;
sf::ThreadLocalPtr<MyClass> objectPtr;

void thread1()
{
    objectPtr = &object1; // doesn't impact thread2
    ...
}

void thread2()
{
    objectPtr = &object2; // doesn't impact thread1
    ...
}

int main()
{
    // Create and launch the two threads
    sf::Thread t1(&thread1);
    sf::Thread t2(&thread2);
    t1.launch();
```

```
    t2.launch();
    return 0;
}
```

`ThreadLocalPtr` is designed for internal use; however you can use it implementation.

Definition at line 41 of file `ThreadLocalPtr.hpp`.

Constructor & Destructor Documentation

```
template<typename T>
sf::ThreadLocalPtr< T >::ThreadLocalPtr ( T * value = NULL )
```

Default constructor.

Parameters

value Optional value to initialize the variable

Member Function Documentation

```
template<typename T>
```

sf::ThreadLocalPtr< T >::operator T *() const

Conversion operator to implicitly convert the pointer to its raw pointer type.

Returns

Pointer to the actual object

```
template<typename T>
```

T& sf::ThreadLocalPtr< T >::operator*() const

Overload of unary operator *.

Like raw pointers, applying the * operator returns a reference to the pointed-to object.

Returns

Reference to the thread-local variable

```
template<typename T>
```

T* sf::ThreadLocalPtr< T >::operator->() const

Overload of operator ->

Similarly to raw pointers, applying the -> operator returns the pointed-to object.

Returns

Pointer to the thread-local variable

```
template<typename T>
```

```
ThreadLocalPtr<T>& sf::ThreadLocalPtr< T >::operator= ( T * value )
```

Assignment operator for a raw pointer parameter.

Parameters

value Pointer to assign

Returns

Reference to self

```
template<typename T>
```

```
ThreadLocalPtr<T>& sf::ThreadLocalPtr< T >::operator= ( const ThreadLocalPtr & right )
```

Assignment operator for a ThreadLocalPtr parameter.

Parameters

right ThreadLocalPtr to assign

Returns

Reference to self

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

- ThreadLocalPtr.hpp

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Static Public Attributes](#) | [Friends](#) | [Related Functions](#) | [List of all members](#)

sf::Time Class Reference

[System module](#)

Represents a time value. [More...](#)

```
#include <Time.hpp>
```

Public Member Functions

`Time ()`

Default constructor. [More...](#)

`float asSeconds () const`

Return the time value as a number of seconds. [More...](#)

`Int32 asMilliseconds () const`

Return the time value as a number of milliseconds. [More...](#)

`Int64 asMicroseconds () const`

Return the time value as a number of microseconds. [More...](#)

Static Public Attributes

static const Time Zero

Predefined "zero" time value. More...

Friends

Time **seconds** (float)

Time **milliseconds** (Int32)

Time **microseconds** (Int64)

Related Functions

(Note that these are not member functions.)

`Time seconds (float amount)`

Construct a time value from a number of seconds. [More...](#)

`Time milliseconds (Int32 amount)`

Construct a time value from a number of milliseconds. [More...](#)

`Time microseconds (Int64 amount)`

Construct a time value from a number of microseconds. [More...](#)

`bool operator== (Time left, Time right)`

Overload of == operator to compare two time values. [More...](#)

`bool operator!= (Time left, Time right)`

Overload of != operator to compare two time values. [More...](#)

`bool operator< (Time left, Time right)`

Overload of < operator to compare two time values. [More...](#)

`bool operator> (Time left, Time right)`

Overload of > operator to compare two time values. [More...](#)

`bool operator<= (Time left, Time right)`

Overload of <= operator to compare two time values. [More...](#)

`bool operator>= (Time left, Time right)`

Overload of >= operator to compare two time values. [More...](#)

`Time operator- (Time right)`

Overload of unary - operator to negate a time value. More...

Time `operator+ (Time left, Time right)`

Overload of binary + operator to add two time values. More...

Time & `operator+= (Time &left, Time right)`

Overload of binary += operator to add/assign two time values. More...

Time `operator- (Time left, Time right)`

Overload of binary - operator to subtract two time values. More...

Time & `operator-= (Time &left, Time right)`

Overload of binary -= operator to subtract/assign two time value

Time `operator* (Time left, float right)`

Overload of binary * operator to scale a time value. More...

Time `operator* (Time left, Int64 right)`

Overload of binary * operator to scale a time value. More...

Time `operator* (float left, Time right)`

Overload of binary * operator to scale a time value. More...

Time `operator* (Int64 left, Time right)`

Overload of binary * operator to scale a time value. More...

Time & `operator*= (Time &left, float right)`

Overload of binary *= operator to scale/assign a time value. More...

Time & `operator*= (Time &left, Int64 right)`

Overload of binary *= operator to scale/assign a time value. More...

Time `operator/ (Time left, float right)`

Overload of binary / operator to scale a time value. More...

`Time operator/ (Time left, Int64 right)`

Overload of binary / operator to scale a time value. More...

`Time & operator/= (Time &left, float right)`

Overload of binary /= operator to scale/assign a time value. Mor

`Time & operator/= (Time &left, Int64 right)`

Overload of binary /= operator to scale/assign a time value. Mor

`float operator/ (Time left, Time right)`

Overload of binary / operator to compute the ratio of two time va

`Time operator% (Time left, Time right)`

Overload of binary % operator to compute remainder of a time v

`Time & operator%= (Time &left, Time right)`

Overload of binary %= operator to compute/assign remainder of

Detailed Description

Represents a time value.

`sf::Time` encapsulates a time value in a flexible way.

It allows to define a time value either as a number of seconds, milliseconds or microseconds. The other way round: you can read a time value as either a number of seconds, milliseconds or microseconds.

By using such a flexible interface, the API doesn't impose any fixed type and let the user choose its own favorite representation.

`Time` values support the usual mathematical operations: you can add or subtract two times, multiply or divide a time by a number, compare two times, etc.

Since they represent a time span and not an absolute time value, times can be added or subtracted.

Usage example:

```
sf::Time t1 = sf::seconds(0.1f);
Int32 milli = t1.asMilliseconds(); // 100

sf::Time t2 = sf::milliseconds(30);
Int64 micro = t2.asMicroseconds(); // 30000

sf::Time t3 = sf::microseconds(-800000);
float sec = t3.asSeconds(); // -0.8

void update(sf::Time elapsed)
{
    position += speed * elapsed.asSeconds();
}

update(sf::milliseconds(100));
```

See also

`sf::Clock`

Definition at line 40 of file `Time.hpp`.

Constructor & Destructor Documentation

sf::Time::Time()

Default constructor.

Sets the time value to zero.

Member Function Documentation

Int64 sf::Time::asMicroseconds () const

Return the time value as a number of microseconds.

Returns

`Time` in microseconds

See also

`asSeconds`, `asMilliseconds`

Int32 sf::Time::asMilliseconds () const

Return the time value as a number of milliseconds.

Returns

`Time` in milliseconds

See also

`asSeconds`, `asMicroseconds`

float sf::Time::asSeconds () const

Return the time value as a number of seconds.

Returns

Time in seconds

See also

[asMilliseconds](#), [asMicroseconds](#)

Friends And Related Function Documentation

Time microseconds (Int64 amount)

Construct a time value from a number of microseconds.

Parameters

amount Number of microseconds

Returns

Time value constructed from the amount of microseconds

See also

seconds, milliseconds

Time milliseconds (Int32 amount)

Construct a time value from a number of milliseconds.

Parameters

amount Number of milliseconds

Returns

Time value constructed from the amount of milliseconds

See also

seconds, microseconds

**bool operator!= (Time left,
Time right
)**

Overload of != operator to compare two time values.

Parameters

left Left operand (a time)

right Right operand (a time)

Returns

True if both time values are different

Time operator% (Time left,
Time right
)

Overload of binary % operator to compute remainder of a time value.

Parameters

left Left operand (a time)

right Right operand (a time)

Returns

left modulo right

Overload of binary %= operator to compute/assign remainder of a time v

Parameters

left Left operand (a time)

right Right operand (a time)

Returns

left modulo right

```
Time operator* ( Time left,  
                  float right  
                )
```

Overload of binary * operator to scale a time value.

Parameters

left Left operand (a time)

right Right operand (a number)

Returns

left multiplied by right

```
Time operator* ( Time left,  
                  Int64 right  
                )
```

Overload of binary * operator to scale a time value.

Parameters

left Left operand (a time)

right Right operand (a number)

Returns

left multiplied by *right*

```
Time operator* ( float left,  
                  Time right  
                )
```

Overload of binary * operator to scale a time value.

Parameters

left Left operand (a number)
right Right operand (a time)

Returns

left multiplied by *right*

```
Time operator* ( Int64 left,  
                  Time right  
                )
```

Overload of binary * operator to scale a time value.

Parameters

left Left operand (a number)
right Right operand (a time)

Returns

left multiplied by *right*

```
Time & operator*=( Time & left,  
                      float    right  
                    )
```

Overload of binary *= operator to scale/assign a time value.

Parameters

left Left operand (a time)
right Right operand (a number)

Returns

left multiplied by *right*

```
Time & operator*=( Time & left,  
                      Int64   right  
                    )
```

Overload of binary *= operator to scale/assign a time value.

Parameters

left Left operand (a time)
right Right operand (a number)

Returns

left multiplied by *right*

```
Time operator+ ( Time left,  
                  Time right  
                )
```

Overload of binary + operator to add two time values.

Parameters

left Left operand (a time)
right Right operand (a time)

Returns

Sum of the two times values

```
Time & operator+=( Time & left,  
                      Time    right  
                    )
```

Overload of binary += operator to add/assign two time values.

Parameters

left Left operand (a time)
right Right operand (a time)

Returns

Sum of the two times values

```
Time operator-( Time right )
```

Overload of unary - operator to negate a time value.

Parameters

right Right operand (a time)

Returns

Opposite of the time value

Time operator- (Time left,
 Time right
)

Overload of binary - operator to subtract two time values.

Parameters

left Left operand (a time)
right Right operand (a time)

Returns

Difference of the two times values

Time & operator-= (Time & left,
 Time right
)

Overload of binary -= operator to subtract/assign two time values.

Parameters

left Left operand (a time)
right Right operand (a time)

Returns

Difference of the two times values

Time operator/ (Time left,

```
float right  
)
```

Overload of binary / operator to scale a time value.

Parameters

left Left operand (a time)
right Right operand (a number)

Returns

left divided by *right*

```
Time operator/ ( Time left,  
                  Int64 right  
)
```

Overload of binary / operator to scale a time value.

Parameters

left Left operand (a time)
right Right operand (a number)

Returns

left divided by *right*

```
float operator/ ( Time left,  
                   Time right  
)
```

Overload of binary / operator to compute the ratio of two time values.

Parameters

left Left operand (a time)
right Right operand (a time)

Returns

left divided by *right*

```
Time & operator/= ( Time & left,  
                     float    right  
                   )
```

Overload of binary /= operator to scale/assign a time value.

Parameters

left Left operand (a time)
right Right operand (a number)

Returns

left divided by *right*

```
Time & operator/= ( Time & left,  
                     Int64    right  
                   )
```

Overload of binary /= operator to scale/assign a time value.

Parameters

left Left operand (a time)
right Right operand (a number)

Returns

left divided by *right*

```
bool operator< ( Time left,  
                  Time right  
                )
```

Overload of < operator to compare two time values.

Parameters

left Left operand (a time)
right Right operand (a time)

Returns

True if *left* is lesser than *right*

```
bool operator<= ( Time left,  
                   Time right  
                 )
```

Overload of <= operator to compare two time values.

Parameters

left Left operand (a time)
right Right operand (a time)

Returns

True if *left* is lesser or equal than *right*

```
bool operator== ( Time left,
```

```
Time right  
)
```

Overload of == operator to compare two time values.

Parameters

left Left operand (a time)
right Right operand (a time)

Returns

True if both time values are equal

```
bool operator> ( Time left,  
                  Time right  
)
```

Overload of > operator to compare two time values.

Parameters

left Left operand (a time)
right Right operand (a time)

Returns

True if *left* is greater than *right*

```
bool operator>= ( Time left,  
                   Time right  
)
```

Overload of >= operator to compare two time values.

Parameters

left Left operand (a time)

right Right operand (a time)

Returns

True if *left* is greater or equal than *right*

Time seconds (float amount)

Construct a time value from a number of seconds.

Parameters

amount Number of seconds

Returns

Time value constructed from the amount of seconds

See also

[milliseconds](#), [microseconds](#)

Member Data Documentation

const Time sf::Time::Zero

Predefined "zero" time value.

Definition at line 85 of file [Time.hpp](#).

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

- [Time.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

`sf::Utf< N >` Class Template Reference

[System module](#)

Utility class providing generic functions for UTF conversions. [More...](#)

```
#include <Utf.hpp>
```

Detailed Description

```
template<unsigned int N>
class sf::Utf< N >
```

Utility class providing generic functions for UTF conversions.

`sf::Utf` is a low-level, generic interface for counting, iterating, encoding and decoding strings. It is able to handle ANSI, wide, latin-1, UTF-8, UTF-16 and UTF-32 strings.

`sf::Utf<X>` functions are all static, these classes are not meant to be instantiated. They are template, so that you can use any character / string type for a given encoding.

It has 3 specializations:

- `sf::Utf<8>` (typedef'd to `sf::Utf8`)
- `sf::Utf<16>` (typedef'd to `sf::Utf16`)
- `sf::Utf<32>` (typedef'd to `sf::Utf32`)

Definition at line 41 of file `Utf.hpp`.

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

- `Utf.hpp`

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Public Attributes](#) | [Related Functions](#) | [List of all members](#)

sf::Vector2< T > Class Template Reference

[System module](#)

Utility template class for manipulating 2-dimensional vectors. [More...](#)

```
#include <Vector2.hpp>
```

Public Member Functions

`Vector2 ()`

Default constructor. [More...](#)

`Vector2 (T X, T Y)`

Construct the vector from its coordinates. [More...](#)

`template<typename U >`

`Vector2 (const Vector2< U > &vector)`

Construct the vector from another type of vector. [More...](#)

Public Attributes

T `x`

X coordinate of the vector. [More...](#)

T `y`

Y coordinate of the vector. [More...](#)

Related Functions

(Note that these are not member functions.)

```
template<typename T>
Vector2< T > operator- (const Vector2< T > &right)
Overload of unary operator -. More...
```

```
template<typename T>
Vector2< T > & operator+= (Vector2< T > &left, const Vector2< T > &rig
Overload of binary operator +=. More...
```

```
template<typename T>
Vector2< T > & operator-= (Vector2< T > &left, const Vector2< T > &rigl
Overload of binary operator -=. More...
```

```
template<typename T>
Vector2< T > operator+ (const Vector2< T > &left, const Vector2< T >
Overload of binary operator +. More...
```

```
template<typename T>
Vector2< T > operator- (const Vector2< T > &left, const Vector2< T >
Overload of binary operator -. More...
```

```
template<typename T>
Vector2< T > operator* (const Vector2< T > &left, T right)
Overload of binary operator *. More...
```

```
template<typename T>
Vector2< T > operator* (T left, const Vector2< T > &right)
Overload of binary operator *. More...
```

```
template<typename T>
Vector2< T > & operator*= (Vector2< T > &left, T right)
Overload of binary operator *=. More...
```

```
template<typename T >
Vector2< T > operator/ (const Vector2< T > &left, T right)
    Overload of binary operator /. More...
```

```
template<typename T >
Vector2< T > & operator/= (Vector2< T > &left, T right)
    Overload of binary operator /=. More...
```

```
template<typename T >
bool operator==(const Vector2< T > &left, const Vector2< T >
    Overload of binary operator ==. More...
```

```
template<typename T >
bool operator!=(const Vector2< T > &left, const Vector2< T >
    Overload of binary operator !=. More...
```

Detailed Description

```
template<typename T>
class sf::Vector2< T >
```

Utility template class for manipulating 2-dimensional vectors.

`sf::Vector2` is a simple class that defines a mathematical vector with two c

It can be used to represent anything that has two dimensions: a size, a po

The template parameter `T` is the type of the coordinates. It can be ϵ operations (`+`, `-`, `/`, `*`) and comparisons (`==`, `!=`), for example `int` or `float`.

You generally don't have to care about the templated form (`sf::\`` specializations have special typedefs:

- `sf::Vector2<float>` is `sf::Vector2f`
- `sf::Vector2<int>` is `sf::Vector2i`
- `sf::Vector2<unsigned int>` is `sf::Vector2u`

The `sf::Vector2` class has a small and simple interface, its `x` and `y` me (there are no accessors like `setX()`, `getX()`) and it contains no mathematic product, length, etc.

Usage example:

```
sf::Vector2f v1(16.5f, 24.f);
v1.x = 18.2f;
float y = v1.y;

sf::Vector2f v2 = v1 * 5.f;
sf::Vector2f v3;
```

```
v3 = v1 + v2;  
bool different = (v2 != v3);
```

Note: for 3-dimensional vectors, see `sf::Vector3`.

Definition at line [37](#) of file `Vector2.hpp`.

Constructor & Destructor Documentation

```
template<typename T>
sf::Vector2< T >::Vector2()
```

Default constructor.

Creates a Vector2(0, 0).

```
template<typename T>
sf::Vector2< T >::Vector2(T X,
                           T Y
                           )
```

Construct the vector from its coordinates.

Parameters

X X coordinate

Y Y coordinate

```
template<typename T>
template<typename U>
sf::Vector2< T >::Vector2(const Vector2< U > & vector)
```

Construct the vector from another type of vector.

This constructor doesn't replace the copy constructor, it's called on constructor will fail to compile if U is not convertible to T.

Parameters

vector Vector to convert

Friends And Related Function Documentation

```
template<typename T >
bool operator!= ( const Vector2< T > & left,
                    const Vector2< T > & right
                )
```

Overload of binary operator !=.

This operator compares strict difference between two vectors.

Parameters

left Left operand (a vector)
right Right operand (a vector)

Returns

True if *left* is not equal to *right*

```
template<typename T >
Vector2< T > operator* ( const Vector2< T > & left,
                                T
                                right
                            )
```

Overload of binary operator *.

Parameters

left Left operand (a vector)
right Right operand (a scalar value)

Returns

Memberwise multiplication by *right*

```
template<typename T >
Vector2< T > operator* ( T left,
                           const Vector2< T > & right
                         )
```

Overload of binary operator *.

Parameters

- left** Left operand (a scalar value)
- right** Right operand (a vector)

Returns

Memberwise multiplication by *left*

```
template<typename T >
Vector2< T > & operator*=( Vector2< T > & left,
                           T right
                         )
```

Overload of binary operator *=.

This operator performs a memberwise multiplication by *right*, and assigns the result back to *left*.

Parameters

- left** Left operand (a vector)
- right** Right operand (a scalar value)

Returns

Reference to *left*

```
template<typename T >
Vector2< T > operator+ ( const Vector2< T > & left,
                           const Vector2< T > & right
)
```

Overload of binary operator +.

Parameters

left Left operand (a vector)

right Right operand (a vector)

Returns

Memberwise addition of both vectors

```
template<typename T >
Vector2< T > & operator+= ( Vector2< T > & left,
                           const Vector2< T > & right
)
```

Overload of binary operator +=.

This operator performs a memberwise addition of both vectors, and assigns the result back to the left operand.

Parameters

left Left operand (a vector)

right Right operand (a vector)

Returns

Reference to *left*

```
template<typename T >
```

Vector2< T > operator- (const Vector2< T > & right)

Overload of unary operator -.

Parameters

right Vector to negate

Returns

Memberwise opposite of the vector

```
template<typename T >
```

**Vector2< T > operator- (const Vector2< T > & left,
 const Vector2< T > & right
)**

Overload of binary operator -.

Parameters

left Left operand (a vector)

right Right operand (a vector)

Returns

Memberwise subtraction of both vectors

```
template<typename T >
```

Vector2< T > & operator-= (Vector2< T > & left,

)

Overload of binary operator /=.

This operator performs a memberwise division by *right*, and assigns the result to *left*.

Parameters

- left** Left operand (a vector)
- right** Right operand (a scalar value)

Returns

Reference to *left*

```
template<typename T >
bool operator==( const Vector2< T > & left,
                  const Vector2< T > & right
                )
```

Overload of binary operator ==.

This operator compares strict equality between two vectors.

Parameters

- left** Left operand (a vector)
- right** Right operand (a vector)

Returns

True if *left* is equal to *right*

Member Data Documentation

```
template<typename T>
```

T sf::Vector2< T >::x

X coordinate of the vector.

Definition at line 75 of file [Vector2.hpp](#).

```
template<typename T>
```

T sf::Vector2< T >::y

Y coordinate of the vector.

Definition at line 76 of file [Vector2.hpp](#).

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

- [Vector2.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Public Attributes](#) | [Related Functions](#) | [List of all members](#)

sf::Vector3< T > Class Template Reference

[System module](#)

Utility template class for manipulating 3-dimensional vectors. [More...](#)

```
#include <Vector3.hpp>
```

Public Member Functions

`Vector3 ()`

Default constructor. [More...](#)

`Vector3 (T X, T Y, T Z)`

Construct the vector from its coordinates. [More...](#)

`template<typename U >`

`Vector3 (const Vector3< U > &vector)`

Construct the vector from another type of vector. [More...](#)

Public Attributes

T `x`

X coordinate of the vector. [More...](#)

T `y`

Y coordinate of the vector. [More...](#)

T `z`

Z coordinate of the vector. [More...](#)

Related Functions

(Note that these are not member functions.)

```
template<typename T>
Vector3< T > operator- (const Vector3< T > &left)
Overload of unary operator -. More...
```

```
template<typename T>
Vector3< T > & operator+= (Vector3< T > &left, const Vector3< T > &rig)
Overload of binary operator +=. More...
```

```
template<typename T>
Vector3< T > & operator-= (Vector3< T > &left, const Vector3< T > &rig)
Overload of binary operator -=. More...
```

```
template<typename T>
Vector3< T > operator+ (const Vector3< T > &left, const Vector3< T >
Overload of binary operator +. More...
```

```
template<typename T>
Vector3< T > operator- (const Vector3< T > &left, const Vector3< T >
Overload of binary operator -. More...
```

```
template<typename T>
Vector3< T > operator* (const Vector3< T > &left, T right)
Overload of binary operator *. More...
```

```
template<typename T>
Vector3< T > operator* (T left, const Vector3< T > &right)
Overload of binary operator *. More...
```

```
template<typename T>
Vector3< T > & operator*= (Vector3< T > &left, T right)
Overload of binary operator *=. More...
```

```
template<typename T >
Vector3< T > operator/ (const Vector3< T > &left, T right)
Overload of binary operator /. More...
```

```
template<typename T >
Vector3< T > & operator/= (Vector3< T > &left, T right)
Overload of binary operator /=. More...
```

```
template<typename T >
bool operator==(const Vector3< T > &left, const Vector3< T >
Overload of binary operator ==. More...
```

```
template<typename T >
bool operator!=(const Vector3< T > &left, const Vector3< T >
Overload of binary operator !=. More...
```

Detailed Description

```
template<typename T>
class sf::Vector3< T >
```

Utility template class for manipulating 3-dimensional vectors.

`sf::Vector3` is a simple class that defines a mathematical vector with three

It can be used to represent anything that has three dimensions: a size, a p

The template parameter `T` is the type of the coordinates. It can be a
operations (+, -, /, *) and comparisons (==, !=), for example int or float.

You generally don't have to care about the templated form (`sf::\`
specializations have special typedefs:

- `sf::Vector3<float>` is `sf::Vector3f`
- `sf::Vector3<int>` is `sf::Vector3i`

The `sf::Vector3` class has a small and simple interface, its x and y me
(there are no accessors like `setX()`, `getX()`) and it contains no mathematic
product, length, etc.

Usage example:

```
sf::Vector3f v1(16.5f, 24.f, -8.2f);
v1.x = 18.2f;
float y = v1.y;
float z = v1.z;

sf::Vector3f v2 = v1 * 5.f;
sf::Vector3f v3;
v3 = v1 + v2;
```

```
bool different = (v2 != v3);
```

Note: for 2-dimensional vectors, see `sf::Vector2`.

Definition at line 37 of file `Vector3.hpp`.

Constructor & Destructor Documentation

```
template<typename T>
sf::Vector3< T >::Vector3()
```

Default constructor.

Creates a Vector3(0, 0, 0).

```
template<typename T>
sf::Vector3< T >::Vector3(T X,
                           T Y,
                           T Z
                           )
```

Construct the vector from its coordinates.

Parameters

X X coordinate

Y Y coordinate

Z Z coordinate

```
template<typename T>
template<typename U >
sf::Vector3< T >::Vector3(const Vector3< U > & vector)
```

Construct the vector from another type of vector.

This constructor doesn't replace the copy constructor, it's called on
constructor will fail to compile if U is not convertible to T.

Parameters

vector Vector to convert

Friends And Related Function Documentation

```
template<typename T >
bool operator!= ( const Vector3< T > & left,
                    const Vector3< T > & right
                )
```

Overload of binary operator !=.

This operator compares strict difference between two vectors.

Parameters

left Left operand (a vector)
right Right operand (a vector)

Returns

True if *left* is not equal to *right*

```
template<typename T >
Vector3< T > operator* ( const Vector3< T > & left,
                                T
                                right
                            )
```

Overload of binary operator *.

Parameters

left Left operand (a vector)
right Right operand (a scalar value)

Returns

Memberwise multiplication by *right*

```
template<typename T >
Vector3< T > operator* ( T left,
                           const Vector3< T > & right
                         )
```

Overload of binary operator *.

Parameters

- left** Left operand (a scalar value)
- right** Right operand (a vector)

Returns

Memberwise multiplication by *left*

```
template<typename T >
Vector3< T > & operator*=( Vector3< T > & left,
                           T right
                         )
```

Overload of binary operator *=.

This operator performs a memberwise multiplication by *right*, and assigns the result back to *left*.

Parameters

- left** Left operand (a vector)
- right** Right operand (a scalar value)

Returns

Reference to *left*

```
template<typename T >
Vector3< T > operator+ ( const Vector3< T > & left,
                           const Vector3< T > & right
)
```

Overload of binary operator +.

Parameters

left Left operand (a vector)

right Right operand (a vector)

Returns

Memberwise addition of both vectors

```
template<typename T >
Vector3< T > & operator+= ( Vector3< T > & left,
                           const Vector3< T > & right
)
```

Overload of binary operator +=.

This operator performs a memberwise addition of both vectors, and assigns the result back to the left operand.

Parameters

left Left operand (a vector)

right Right operand (a vector)

Returns

Reference to *left*

```
template<typename T >
Vector3< T > operator- ( const Vector3< T > & left )
```

Overload of unary operator -.

Parameters

left Vector to negate

Returns

Memberwise opposite of the vector

```
template<typename T >
Vector3< T > operator- ( const Vector3< T > & left,
                           const Vector3< T > & right
                         )
```

Overload of binary operator -.

Parameters

left Left operand (a vector)

right Right operand (a vector)

Returns

Memberwise subtraction of both vectors

```
template<typename T >
Vector3< T > & operator-= ( Vector3< T > & left,
```


)

Overload of binary operator /=.

This operator performs a memberwise division by *right*, and assigns the result to *left*.

Parameters

- left** Left operand (a vector)
- right** Right operand (a scalar value)

Returns

Reference to *left*

```
template<typename T >  
bool operator==( const Vector3< T > & left,  
                  const Vector3< T > & right  
                )
```

Overload of binary operator ==.

This operator compares strict equality between two vectors.

Parameters

- left** Left operand (a vector)
- right** Right operand (a vector)

Returns

True if *left* is equal to *right*

Member Data Documentation

```
template<typename T>
```

T sf::Vector3< T >::x

X coordinate of the vector.

Definition at line [76](#) of file [Vector3.hpp](#).

```
template<typename T>
```

T sf::Vector3< T >::y

Y coordinate of the vector.

Definition at line [77](#) of file [Vector3.hpp](#).

```
template<typename T>
```

T sf::Vector3< T >::z

Z coordinate of the vector.

Definition at line [78](#) of file [Vector3.hpp](#).

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

- [Vector3.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

[Classes](#) | [Typedefs](#) | [Enumerations](#)

Window module

Provides OpenGL-based windows, and abstractions for events and input handling.

Classes

class **sf::Context**

Class holding a valid drawing context. [More...](#)

class **sf::ContextSettings**

Structure defining the settings of the OpenGL context attached to

class **sf::Event**

Defines a system event and its parameters. [More...](#)

class **sf::GLResource**

Base class for classes that require an OpenGL context. [More...](#)

class **sf::Joystick**

Give access to the real-time state of the joysticks. [More...](#)

class **sf::Keyboard**

Give access to the real-time state of the keyboard. [More...](#)

class **sf::Mouse**

Give access to the real-time state of the mouse. [More...](#)

class **sf::Sensor**

Give access to the real-time state of the sensors. [More...](#)

class **sf::Touch**

Give access to the real-time state of the touches. [More...](#)

class **sf::VideoMode**

VideoMode defines a video mode (width, height, bpp) [More...](#)

class **sf::Window**

Window that serves as a target for OpenGL rendering. More...

Typedefs

typedef platform-specific `sf::WindowHandle`

Define a low-level window handle type, specific

Enumerations

```
{  
    sf::Style::None = 0, sf::Style::Titlebar = 1 << 0, sf::Style::Resize  
enum 2,  
    sf::Style::Fullscreen = 1 << 3, sf::Style::Default = Titlebar | Resi  
}  
Enumeration of the window styles. More...
```

Detailed Description

Provides OpenGL-based windows, and abstractions for events and input |

Typedef Documentation

sf::WindowHandle

Define a low-level window handle type, specific to each platform.

Platform	Type
Windows	HWND
Linux/FreeBSD	Window
Mac OS X	either <code>NSWindow*</code> or <code>NSView*</code> , disguised as <code>void*</code>
iOS	<code>UIWindow*</code>
Android	<code>ANativeWindow*</code>

Mac OS X Specification

On Mac OS X, a `sf::Window` can be created either from an existing `NSWindow`. If a `sf::Window` is created from a window, SFML will use its content. `sf::Window::getSystemHandle()` will return the handle that was used to create the `NSWindow*` by default.

Definition at line 68 of file `WindowHandle.hpp`.

Enumeration Type Documentation

anonymous enum

Enumeration of the window styles.

Enumerator	
None	No border / title bar (this flag and all others are mutually exclusive).
Titlebar	Title bar + fixed border.
Resize	Title bar + resizable border + maximize button.
Close	Title bar + close button.
Fullscreen	Fullscreen mode (this flag and all others are mutually exclusive).
Default	Default window style.

Definition at line 38 of file [WindowStyle.hpp](#).

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Static Public Member Functions](#) | [Static Private Member Functions](#) | [List of all members](#)

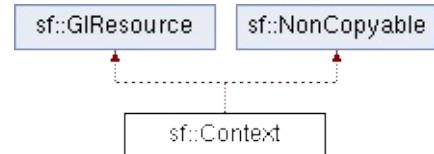
sf::Context Class Reference

[Window module](#)

Class holding a valid drawing context. [More...](#)

```
#include <Context.hpp>
```

Inheritance diagram for sf::Context:



Public Member Functions

`Context ()`
Default constructor. [More...](#)

`~Context ()`
Destructor. [More...](#)

`bool setActive (bool active)`
Activate or deactivate explicitly the context. [More...](#)

`const ContextSettings & getSettings () const`
Get the settings of the context. [More...](#)

`Context (const ContextSettings &settings, unsigned height)`
Construct a in-memory context. [More...](#)

Static Public Member Functions

static bool `isExtensionAvailable` (const char *name)
Check whether a given OpenGL extension is available.

static GLFunctionPointer `getFunction` (const char *name)
Get the address of an OpenGL function. [More..](#)

static const Context * `getActiveContext` ()
Get the currently active context. [More...](#)

Static Private Member Functions

```
static void ensureGLContext ()
```

Empty function for ABI compatibility, use acquireTransientContext()

Detailed Description

Class holding a valid drawing context.

If you need to make OpenGL calls without having an active window (like in a thread), you can create an instance of this class to get a valid context.

Having a valid context is necessary for *every* OpenGL call.

Note that a context is only active in its current thread, if you create a new thread, it will have its own context by default.

To use a `sf::Context` instance, just construct it and let it live as long as you need it. As soon as its activation is needed, all it has to do is to exist. Its destructor will take care of releasing all attached resources.

Usage example:

```
void threadFunction(void*)
{
    sf::Context context;
    // from now on, you have a valid context

    // you can make OpenGL calls
    glClear(GL_DEPTH_BUFFER_BIT);
}
// the context is automatically deactivated and destroyed
// by the sf::Context destructor
```

Definition at line 50 of file `Context.hpp`.

Constructor & Destructor Documentation

`sf::Context::Context()`

Default constructor.

The constructor creates and activates the context

`sf::Context::~Context()`

Destructor.

The destructor deactivates and destroys the context

```
sf::Context::Context ( const ContextSettings & settings,  
                      unsigned int width,  
                      unsigned int height  
)
```

Construct a in-memory context.

This constructor is for internal use, you don't need to bother with it.

Parameters

settings Creation parameters

width Back buffer width

height Back buffer height

Member Function Documentation

static const Context* sf::Context::getActiveContext()

Get the currently active context.

Returns

The currently active context or NULL if none is active

static GIFunctionPointer sf::Context::getFunction(const char * name)

Get the address of an OpenGL function.

Parameters

name Name of the function to get the address of

Returns

Address of the OpenGL function, 0 on failure

const ContextSettings& sf::Context::getSettings() const

Get the settings of the context.

Note that these settings may be different than the ones passed to them, adjusted if the original settings are not directly supported by the system.

Returns

Structure containing the settings

static bool sf::Context::isExtensionAvailable (const char * name)

Check whether a given OpenGL extension is available.

Parameters

name Name of the extension to check for

Returns

True if available, false if unavailable

bool sf::Context::setActive (bool active)

Activate or deactivate explicitly the context.

Parameters

active True to activate, false to deactivate

Returns

True on success, false on failure

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

- [Context.hpp](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Types](#) | [Public Member Functions](#) | [Public Attributes](#) | [List of all members](#)

sf::ContextSettings Class Reference

[Window module](#)

Structure defining the settings of the OpenGL context attached to a window.

```
#include <ContextSettings.hpp>
```

Public Types

```
enum Attribute { Default = 0, Core = 1 << 0, Debug = 1 << 2 }
```

Enumeration of the context attribute flags. [More...](#)

Public Member Functions

`ContextSettings` (unsigned int depth=0, unsigned int stencil=0, unsigned int major=1, unsigned int minor=1, unsigned int attributes=`Default`, bool skipValidation=false)

Default constructor. [More...](#)

Public Attributes

`unsigned int depthBits`

Bits of the depth buffer. [More...](#)

`unsigned int stencilBits`

Bits of the stencil buffer. [More...](#)

`unsigned int antialiasingLevel`

Level of antialiasing. [More...](#)

`unsigned int majorVersion`

Major number of the context version to create. [More...](#)

`unsigned int minorVersion`

Minor number of the context version to create. [More...](#)

`Uint32 attributeFlags`

The attribute flags to create the context with. [More...](#)

`bool sRgbCapable`

Whether the context framebuffer is sRGB capable. [More...](#)

Detailed Description

Structure defining the settings of the OpenGL context attached to a window.

`ContextSettings` allows to define several advanced settings of the OpenGL context.

All these settings with the exception of the compatibility flag and anti-aliasing level are used for regular SFML rendering (graphics module), so you may need to use this structure as a windowing system for custom OpenGL rendering.

The `depthBits` and `stencilBits` members define the number of bits per pixel for depth and stencil buffers.

`antialiasingLevel` represents the requested number of multisampling levels.

`majorVersion` and `minorVersion` define the version of the OpenGL context. Versions greater or equal to 3.0 are relevant; versions lesser than 3.0 are all handled as any version < 3.0 if you don't want an OpenGL 3 context).

When requesting a context with a version greater or equal to 3.2, you have to specify the core flag. The context should follow the core or compatibility profile of all newer (>= 3.2) OpenGL contexts. In versions 3.0 and 3.1 there is only the core profile. By default a compatibility profile is selected. You need to specify the core flag if you want a core profile context to use vertex arrays.

Warning: The graphics module will not function if you request a compatibility profile context and the core flag is disabled. The core flag is enabled by default if the major version is 3.2 or higher.

Attributes: The attributes are set to Default if you want to use the graphics module.

Setting the `debug` attribute flag will request a context with additional OpenGL extensions. Depending on the system, this might be required for advanced OpenGL features. This flag is disabled by default.

Special Note for OS X: Apple only supports choosing between either a compatibility profile or a core profile. If you request a compatibility profile, the OpenGL version will be set to 2.1. If you request a core profile, the OpenGL version will be set to 3.2.

core context (OpenGL version depends on the operating system version contexts are not supported. Further information is available on the [OpenGL Context](#) page). Note that OpenGL ES contexts are not supported. Also, OpenGL ES 2.0 contexts also currently does not support debug contexts.

Please note that these values are only a hint. No failure will be reported if the requested OpenGL version is not supported by the system; instead, SFML will try to find the closest valid OpenGL version supported by the system, based on the settings that the window actually used to create its context, with Windows and Mac OS X being the most permissive.

Definition at line 36 of file [ContextSettings.hpp](#).

Member Enumeration Documentation

enum sf::ContextSettings::Attribute

Enumeration of the context attribute flags.

Enumerator	
Default	Non-debug, compatibility context (this and the core attribute are set).
Core	Core attribute.
Debug	Debug attribute.

Definition at line 42 of file [ContextSettings.hpp](#).

Constructor & Destructor Documentation

```
sf::ContextSettings::ContextSettings ( unsigned int depth = 0,  
                                     unsigned int stencil = 0,  
                                     unsigned int antialiasing = 0,  
                                     unsigned int major = 1,  
                                     unsigned int minor = 1,  
                                     unsigned int attributes = Defa  
                                     bool sRgb = false  
                               )
```

Default constructor.

Parameters

depth	Depth buffer bits
stencil	Stencil buffer bits
antialiasing	Antialiasing level
major	Major number of the context version
minor	Minor number of the context version
attributes	Attribute flags of the context
sRgb	sRGB capable framebuffer

Definition at line 61 of file [ContextSettings.hpp](#).

Member Data Documentation

unsigned int sf::ContextSettings::antialiasingLevel

Level of antialiasing.

Definition at line [77](#) of file [ContextSettings.hpp](#).

Uint32 sf::ContextSettings::attributeFlags

The attribute flags to create the context with.

Definition at line [80](#) of file [ContextSettings.hpp](#).

unsigned int sf::ContextSettings::depthBits

Bits of the depth buffer.

Definition at line [75](#) of file [ContextSettings.hpp](#).

unsigned int sf::ContextSettings::majorVersion

Major number of the context version to create.

Definition at line [78](#) of file [ContextSettings.hpp](#).

unsigned int sf::ContextSettings::minorVersion

Minor number of the context version to create.

Definition at line [79](#) of file [ContextSettings.hpp](#).

bool sf::ContextSettings::sRgbCapable

Whether the context framebuffer is sRGB capable.

Definition at line [81](#) of file [ContextSettings.hpp](#).

unsigned int sf::ContextSettings::stencilBits

Bits of the stencil buffer.

Definition at line [76](#) of file [ContextSettings.hpp](#).

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

- [ContextSettings.hpp](#)
-

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Classes](#) | [Public Types](#) | [Public Attributes](#) | [List of all members](#)

sf::Event Class Reference

[Window module](#)

Defines a system event and its parameters. [More...](#)

```
#include <Event.hpp>
```

Classes

struct **JoystickButtonEvent**

Joystick buttons events parameters (JoystickButtonPressed, JoystickButtonReleased)

struct **JoystickConnectEvent**

Joystick connection events parameters (JoystickConnected, JoystickDisconnected)

struct **JoystickMoveEvent**

Joystick axis move event parameters (JoystickMoved) [More...](#)

struct **KeyEvent**

Keyboard event parameters (KeyPressed, KeyReleased) [More...](#)

struct **MouseButtonEvent**

Mouse buttons events parameters (MouseButtonPressed, MouseButtonReleased)

struct **MouseMoveEvent**

Mouse move event parameters (MouseMoved) [More...](#)

struct **MouseWheelEvent**

Mouse wheel events parameters (MouseWheelMoved) [More...](#)

struct **MouseWheelScrollView**

Mouse wheel events parameters (MouseWheelScrolled) [More...](#)

struct **SensorEvent**

Sensor event parameters (SensorChanged) [More...](#)

struct **SizeEvent**

Size events parameters (Resized) [More...](#)

struct **TextEvent**

Text event parameters (TextEntered) [More...](#)

struct **TouchEvent**

Touch events parameters (TouchBegan, TouchMoved, TouchEnd)

Public Types

```
enum EventType {  
    Closed, Resized, LostFocus, GainedFocus,  
    TextEntered, KeyPressed, KeyReleased, MouseWheelMoved,  
    MouseWheelScrolled, MouseButtonPressed, MouseButtonRele  
    MouseEntered, MouseLeft, JoystickButtonPressed, JoystickBu  
    JoystickMoved, JoystickConnected, JoystickDisconnected, Tou  
    TouchMoved, TouchEnded, SensorChanged, Count  
}
```

Enumeration of the different types of events. [More...](#)

Public Attributes

	EventType type Type of the event. More
union { SizeEvent size	Size event parameters (
KeyEvent key	Key event parameters (I Event::KeyReleased) M
TextEvent text	Text event parameters (
MouseMoveEvent mouseMove	Mouse move event para More...
MouseButtonEvent mouseButton	Mouse button event par (Event::MouseButtonPre Event::MouseButtonRel
MouseWheelEvent mouseWheel	Mouse wheel event para (Event::MouseWheelMo
MouseWheelScrollEvent mouseWheelScroll	Mouse wheel event para (Event::MouseWheelSci

```
JoystickMoveEvent joystickMove
```

Joystick move event parameter
More...

```
JoystickButtonEvent joystickButton
```

Joystick button event parameter
(Event::JoystickButtonParameter
Event::JoystickButtonResult)

```
JoystickConnectEvent joystickConnect
```

Joystick (dis)connect event parameter
(Event::JoystickConnectParameter
Event::JoystickDisconnectParameter)

```
TouchEvent touch
```

Touch events parameter
Event::TouchMoved, Event::TouchDown, Event::TouchUp

```
SensorEvent sensor
```

Sensor event parameter
More...

```
};
```

Detailed Description

Defines a system event and its parameters.

`sf::Event` holds all the informations about a system event that just happen

Events are retrieved using the `sf::Window::pollEvent` and `sf::Window::waitForEvent`.

A `sf::Event` instance contains the type of the event (mouse moved, key pressed, etc.) as well as the details about this particular event. Please note that the event parameters are not initialized by default which means that only the member matching the type of the event will be initialized while others will have undefined values and must not be read if the type of the event cannot be matched. For example, if you received a KeyPressed event, then you must read the `event.key` member while `event.MouseMove` or `event.text` will have undefined values.

Usage example:

```
sf::Event event;
while (window.pollEvent(event))
{
    // Request for closing the window
    if (event.type == sf::Event::Closed)
        window.close();

    // The escape key was pressed
    if ((event.type == sf::Event::KeyPressed) && (event.key.code == sf::Keyboard::Escape))
        window.close();

    // The window was resized
    if (event.type == sf::Event::Resized)
        doSomethingWithTheNewSize(event.size.width, event.size.height);

    // etc ...
}
```

Definition at line 44 of file `Event.hpp`.

Member Enumeration Documentation

enum sf::Event::EventType

Enumeration of the different types of events.

Enumerator	
Closed	The window requested to be closed (no data)
Resized	The window was resized (data in event.size)
LostFocus	The window lost the focus (no data)
GainedFocus	The window gained the focus (no data)
TextEntered	A character was entered (data in event.text)
KeyPressed	A key was pressed (data in event.key)
KeyReleased	A key was released (data in event.key)
MouseWheelMoved	The mouse wheel was scrolled (data in event.mouseWheel)
MouseWheelScrolled	

	The mouse wheel was scrolled (data in event.mousewheel)
MouseButtonPressed	A mouse button was pressed (data in event.button)
MouseButtonReleased	A mouse button was released (data in event.button)
MouseMoved	The mouse cursor moved (data in event.mouse)
MouseEntered	The mouse cursor entered the area of the window
MouseLeft	The mouse cursor left the area of the window
JoystickButtonPressed	A joystick button was pressed (data in event.joystickbutton)
JoystickButtonReleased	A joystick button was released (data in event.joystickbutton)
JoystickMoved	The joystick moved along an axis (data in event.joystick)
JoystickConnected	A joystick was connected (data in event.joystick)
JoystickDisconnected	A joystick was disconnected (data in event.joystick)
TouchBegan	A touch event began (data in event.touch)
TouchMoved	A touch moved (data in event.touch)
TouchEnded	

	A touch event ended (data in event.touch)
SensorChanged	A sensor value changed (data in event.sensc)
Count	Keep last – the total number of event types.

Definition at line [187](#) of file Event.hpp.

Member Data Documentation

JoystickButtonEvent sf::Event::joystickButton

Joystick button event parameters (Event::JoystickButtonPressed, Event::JoystickButtonReleased)

Definition at line 231 of file Event.hpp.

JoystickConnectEvent sf::Event::joystickConnect

Joystick (dis)connect event parameters (Event::JoystickConnected, Event::JoystickDisconnected)

Definition at line 232 of file Event.hpp.

JoystickMoveEvent sf::Event::joystickMove

Joystick move event parameters (Event::JoystickMoved)

Definition at line 230 of file Event.hpp.

KeyEvent sf::Event::key

Key event parameters (Event::KeyPressed, Event::KeyReleased)

Definition at line 224 of file Event.hpp.

MouseButtonEvent sf::Event::mouseButton

Mouse button event parameters ([Event::MouseButtonPressed](#), [Event::MouseButtonReleased](#))

Definition at line 227 of file [Event.hpp](#).

MouseMoveEvent sf::Event::mouseMove

Mouse move event parameters ([Event::MouseMoved](#))

Definition at line 226 of file [Event.hpp](#).

MouseWheelEvent sf::Event::mouseWheel

Mouse wheel event parameters ([Event::MouseWheelMoved](#)) (deprecate [sf::Event::mouseWheel](#))

Definition at line 228 of file [Event.hpp](#).

MouseWheelScrollEvent sf::Event::mouseWheelScroll

Mouse wheel event parameters ([Event::MouseWheelScrolled](#))

Definition at line 229 of file [Event.hpp](#).

SensorEvent sf::Event::sensor

Sensor event parameters (`Event::SensorChanged`)

Definition at line 234 of file `Event.hpp`.

SizeEvent sf::Event::size

Size event parameters (`Event::Resized`)

Definition at line 223 of file `Event.hpp`.

TextEvent sf::Event::text

Text event parameters (`Event::TextEntered`)

Definition at line 225 of file `Event.hpp`.

TouchEvent sf::Event::touch

Touch events parameters (`Event::TouchBegan`, `Event::TouchMoved`, `Event::TouchReleased`)

Definition at line 233 of file `Event.hpp`.

EventType sf::Event::type

Type of the event.

Definition at line 219 of file `Event.hpp`.

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

- [Event.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#).

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::JoystickButtonEvent Struct Reference

Joystick buttons events parameters (JoystickButtonPressed, JoystickButtonReleased)

#include <Event.hpp>

Public Attributes

`unsigned int joystickId`

Index of the joystick (in range [0 .. Joystick::Count - 1]) More...

`unsigned int button`

Index of the button that has been pressed (in range [0 .. Joystick::Buttons - 1]) More...

Detailed Description

Joystick buttons events parameters (JoystickButtonPressed, JoystickButtonReleased).

Definition at line 154 of file [Event.hpp](#).

Member Data Documentation

unsigned int sf::Event::JoystickButtonEvent::button

Index of the button that has been pressed (in range [0 .. Joystick::Button

Definition at line [157](#) of file [Event.hpp](#).

unsigned int sf::Event::JoystickButtonEvent::joystickId

Index of the joystick (in range [0 .. Joystick::Count - 1])

Definition at line [156](#) of file [Event.hpp](#).

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

- [Event.hpp](#)

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::JoystickConnectEvent Struct Reference

Joystick connection events parameters (JoystickConnected, JoystickDisconnected)

#include <Event.hpp>

Public Attributes

unsigned int **joystickId**

Index of the joystick (in range [0 .. Joystick::Count - 1]) More...

Detailed Description

Joystick connection events parameters (JoystickConnected, JoystickDiscor

Definition at line 133 of file [Event.hpp](#).

Member Data Documentation

unsigned int sf::Event::JoystickConnectEvent::joystickId

Index of the joystick (in range [0 .. Joystick::Count - 1])

Definition at line 135 of file Event.hpp.

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

- Event.hpp
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::JoystickMoveEvent Struct Reference

Joystick axis move event parameters (JoystickMoved) [More...](#)

```
#include <Event.hpp>
```

Public Attributes

unsigned int **joystickId**

Index of the joystick (in range [0 .. Joystick::Count - 1]) [M](#)

Joystick::Axis axis

Axis on which the joystick moved. [More...](#)

float **position**

New position on the axis (in range [-100 .. 100]) [More...](#)

Detailed Description

Joystick axis move event parameters (JoystickMoved)

Definition at line 142 of file [Event.hpp](#).

Member Data Documentation

Joystick::Axis sf::Event::JoystickMoveEvent::axis

Axis on which the joystick moved.

Definition at line [145](#) of file Event.hpp.

unsigned int sf::Event::JoystickMoveEvent::joystickId

Index of the joystick (in range [0 .. Joystick::Count - 1])

Definition at line [144](#) of file Event.hpp.

float sf::Event::JoystickMoveEvent::position

New position on the axis (in range [-100 .. 100])

Definition at line [146](#) of file Event.hpp.

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

- [Event.hpp](#)
-

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::KeyEvent Struct Reference

Keyboard event parameters (KeyPressed, KeyReleased) [More...](#)

#include <Event.hpp>

Public Attributes

Keyboard::Key **code**

Code of the key that has been pressed. [More...](#)

bool **alt**

Is the Alt key pressed? [More...](#)

bool **control**

Is the Control key pressed? [More...](#)

bool **shift**

Is the Shift key pressed? [More...](#)

bool **system**

Is the System key pressed? [More...](#)

Detailed Description

Keyboard event parameters (KeyPressed, KeyReleased)

Definition at line [62](#) of file [Event.hpp](#).

Member Data Documentation

bool sf::Event::KeyEvent::alt

Is the Alt key pressed?

Definition at line [65](#) of file [Event.hpp](#).

Keyboard::Key sf::Event::KeyEvent::code

Code of the key that has been pressed.

Definition at line [64](#) of file [Event.hpp](#).

bool sf::Event::KeyEvent::control

Is the Control key pressed?

Definition at line [66](#) of file [Event.hpp](#).

bool sf::Event::KeyEvent::shift

Is the Shift key pressed?

Definition at line [67](#) of file [Event.hpp](#).

bool sf::Event::KeyEvent::system

Is the System key pressed?

Definition at line **68** of file [Event.hpp](#).

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

- [Event.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

Classes

[Files](#)

[Class List](#)

[Class Index](#)

[Class Hierarchy](#)

[Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::MouseEvent Struct Reference

Mouse buttons events parameters (MouseButtonPressed, MouseButtonR

```
#include <Event.hpp>
```

Public Attributes

`Mouse::Button button`

Code of the button that has been pressed. [More...](#)

`int x`

X position of the mouse pointer, relative to the left of the

`int y`

Y position of the mouse pointer, relative to the top of the

Detailed Description

Mouse buttons events parameters (MouseButtonPressed, MouseButtonR

Definition at line 95 of file [Event.hpp](#).

Member Data Documentation

Mouse::Button sf::Event::MouseButtonEvent::button

Code of the button that has been pressed.

Definition at line [97](#) of file [Event.hpp](#).

int sf::Event::MouseButtonEvent::x

X position of the mouse pointer, relative to the left of the owner window.

Definition at line [98](#) of file [Event.hpp](#).

int sf::Event::MouseButtonEvent::y

Y position of the mouse pointer, relative to the top of the owner window.

Definition at line [99](#) of file [Event.hpp](#).

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

- [Event.hpp](#)
-

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::MouseMoveEvent Struct Reference

Mouse move event parameters (MouseMoved) [More...](#)

```
#include <Event.hpp>
```

Public Attributes

int **x**

X position of the mouse pointer, relative to the left of the owner window

int **y**

Y position of the mouse pointer, relative to the top of the owner window

Detailed Description

Mouse move event parameters (MouseMoved)

Definition at line [84](#) of file [Event.hpp](#).

Member Data Documentation

int sf::Event::MouseMoveEvent::x

X position of the mouse pointer, relative to the left of the owner window.

Definition at line [86](#) of file [Event.hpp](#).

int sf::Event::MouseMoveEvent::y

Y position of the mouse pointer, relative to the top of the owner window.

Definition at line [87](#) of file [Event.hpp](#).

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

- [Event.hpp](#)

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

Classes

[Files](#)

[Class List](#)

[Class Index](#)

[Class Hierarchy](#)

[Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::MouseWheelEvent Struct Reference

Mouse wheel events parameters (MouseWheelMoved) [More...](#)

#include <Event.hpp>

Public Attributes

`int delta`

Number of ticks the wheel has moved (positive is up, negative is down)

`int x`

X position of the mouse pointer, relative to the left of the owner window

`int y`

Y position of the mouse pointer, relative to the top of the owner window

Detailed Description

Mouse wheel events parameters (MouseWheelMoved)

Deprecated:

This event is deprecated and potentially inaccurate. Use MouseWheelMoved instead.

Definition at line 109 of file [Event.hpp](#).

Member Data Documentation

int sf::Event::MouseWheelEvent::delta

Number of ticks the wheel has moved (positive is up, negative is down)

Definition at line [111](#) of file [Event.hpp](#).

int sf::Event::MouseWheelEvent::x

X position of the mouse pointer, relative to the left of the owner window.

Definition at line [112](#) of file [Event.hpp](#).

int sf::Event::MouseWheelEvent::y

Y position of the mouse pointer, relative to the top of the owner window.

Definition at line [113](#) of file [Event.hpp](#).

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

- [Event.hpp](#)
-

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::MouseWheelScrollEvent Struct Reference

Mouse wheel events parameters (MouseWheelScrolled) [More...](#)

```
#include <Event.hpp>
```

Public Attributes

`Mouse::Wheel wheel`

Which wheel (for mice with multiple ones) [More...](#)

`float delta`

Wheel offset (positive is up/left, negative is down/right). For non-integral offsets. [More...](#)

`int x`

X position of the mouse pointer, relative to the left of the

`int y`

Y position of the mouse pointer, relative to the top of the

Detailed Description

Mouse wheel events parameters (MouseWheelScrolled)

Definition at line 120 of file [Event.hpp](#).

Member Data Documentation

float sf::Event::MouseWheelScrollEvent::delta

Wheel offset (positive is up/left, negative is down/right). High-precision offsets.

Definition at line 123 of file Event.hpp.

Mouse::Wheel sf::Event::MouseWheelScrollEvent::wheel

Which wheel (for mice with multiple ones)

Definition at line 122 of file Event.hpp.

int sf::Event::MouseWheelScrollEvent::x

X position of the mouse pointer, relative to the left of the owner window.

Definition at line 124 of file Event.hpp.

int sf::Event::MouseWheelScrollEvent::y

Y position of the mouse pointer, relative to the top of the owner window.

Definition at line 125 of file Event.hpp.

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

- [Event.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::SensorEvent Struct Reference

Sensor event parameters (SensorChanged) [More...](#)

```
#include <Event.hpp>
```

Public Attributes

`Sensor::Type type`

Type of the sensor. More...

`float x`

Current value of the sensor on X axis. More...

`float y`

Current value of the sensor on Y axis. More...

`float z`

Current value of the sensor on Z axis. More...

Detailed Description

Sensor event parameters (SensorChanged)

Definition at line 175 of file `Event.hpp`.

Member Data Documentation

Sensor::Type sf::Event::SensorEvent::type

Type of the sensor.

Definition at line 177 of file Event.hpp.

float sf::Event::SensorEvent::x

Current value of the sensor on X axis.

Definition at line 178 of file Event.hpp.

float sf::Event::SensorEvent::y

Current value of the sensor on Y axis.

Definition at line 179 of file Event.hpp.

float sf::Event::SensorEvent::z

Current value of the sensor on Z axis.

Definition at line 180 of file Event.hpp.

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

- Event.hpp
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::SizeEvent Struct Reference

Size events parameters (Resized) [More...](#)

#include <Event.hpp>

Public Attributes

`unsigned int width`

New width, in pixels. More...

`unsigned int height`

New height, in pixels. More...

Detailed Description

Size events parameters (Resized)

Definition at line [52](#) of file [Event.hpp](#).

Member Data Documentation

unsigned int sf::Event::SizeEvent::height

New height, in pixels.

Definition at line 55 of file [Event.hpp](#).

unsigned int sf::Event::SizeEvent::width

New width, in pixels.

Definition at line 54 of file [Event.hpp](#).

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

- [Event.hpp](#)
-

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::TextEvent Struct Reference

Text event parameters (TextEntered) [More...](#)

```
#include <Event.hpp>
```

Public Attributes

Uint32 `unicode`

UTF-32 Unicode value of the character. More...

Detailed Description

`Text` event parameters (`TextEntered`)

Definition at line 75 of file `Event.hpp`.

Member Data Documentation

Uint32 sf::Event::TextEvent::unicode

UTF-32 Unicode value of the character.

Definition at line [77](#) of file [Event.hpp](#).

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

- [Event.hpp](#)
-

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

Classes

[Files](#)

[Class List](#)

[Class Index](#)

[Class Hierarchy](#)

[Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Event::TouchEvent Struct Reference

Touch events parameters (TouchBegan, TouchMoved, TouchEnded) [More...](#)

#include <Event.hpp>

Public Attributes

unsigned int **finger**

Index of the finger in case of multi-touch events. More...

int **x**

X position of the touch, relative to the left of the owner window.

int **y**

Y position of the touch, relative to the top of the owner window.

Detailed Description

Touch events parameters (TouchBegan, TouchMoved, TouchEnded)

Definition at line 164 of file `Event.hpp`.

Member Data Documentation

unsigned int sf::Event::TouchEvent::finger

Index of the finger in case of multi-touch events.

Definition at line [166](#) of file [Event.hpp](#).

int sf::Event::TouchEvent::x

X position of the touch, relative to the left of the owner window.

Definition at line [167](#) of file [Event.hpp](#).

int sf::Event::TouchEvent::y

Y position of the touch, relative to the top of the owner window.

Definition at line [168](#) of file [Event.hpp](#).

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

- [Event.hpp](#)
-

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Classes](#) | [Protected Member Functions](#) | [Static Protected Member Functions](#) | [List of all members](#)

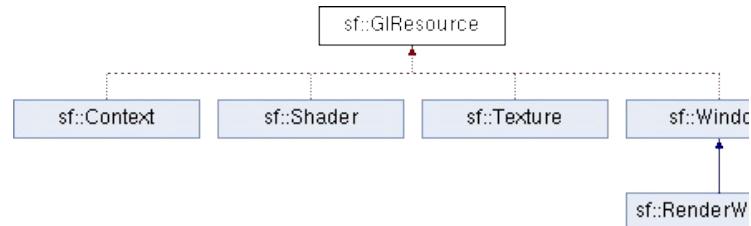
sf::GlResource Class Reference

Window module

Base class for classes that require an OpenGL context. More...

```
#include <GlResource.hpp>
```

Inheritance diagram for sf::GlResource:



Classes

class **TransientContextLock**

RAlI helper class to temporarily lock an available context for use.

Protected Member Functions

`GIResource ()`

Default constructor. More...

`~GIResource ()`

Destructor. More...

Static Protected Member Functions

```
static void ensureGLContext ()
```

Empty function for ABI compatibility, use acquireTransientContext

Detailed Description

Base class for classes that require an OpenGL context.

This class is for internal use only, it must be the base of every class that requires an OpenGL context in order to work.

Definition at line 44 of file [GIResource.hpp](#).

Constructor & Destructor Documentation

sf::GIResource::GIResource ()

Default constructor.

sf::GIResource::~GIResource ()

Destructor.

Member Function Documentation

static void sf::GIResource::ensureGIContext()

Empty function for ABI compatibility, use acquireTransientContext instead.

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

- [GIResource.hpp](#)

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

Classes

[Files](#)

[Class List](#)

[Class Index](#)

[Class Hierarchy](#)

[Class Members](#)

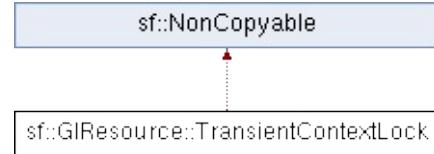
[Public Member Functions](#) | [List of all members](#)

sf::GlResource::TransientContextLock Class

RAII helper class to temporarily lock an available context for use. [More...](#)

```
#include <GlResource.hpp>
```

Inheritance diagram for sf::GlResource::TransientContextLock:



Public Member Functions

`TransientContextLock ()`

Default constructor. More...

`~TransientContextLock ()`

Destructor. More...

Detailed Description

RAII helper class to temporarily lock an available context for use.

Definition at line [70](#) of file [GIResource.hpp](#).

Constructor & Destructor Documentation

sf::GIResource::TransientContextLock::TransientContextLock()

Default constructor.

sf::GIResource::TransientContextLock::~TransientContextLock()

Destructor.

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

- [GIResource.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Classes](#) | [Public Types](#) | [Static Public Member Functions](#) | [List of all members](#)

sf::Joystick Class Reference

[Window module](#)

Give access to the real-time state of the joysticks. [More...](#)

```
#include <Joystick.hpp>
```

Classes

struct **Identification**

Structure holding a joystick's identification. [More...](#)

Public Types

```
enum { Count = 8, ButtonCount = 32, AxisCount = 8 }
```

Constants related to joysticks capabilities. [More...](#)

```
enum Axis {
    X, Y, Z, R,
    U, V, PovX, PovY
}
```

Axes supported by SFML joysticks. [More...](#)

Static Public Member Functions

static bool **isConnected** (unsigned int joystick)
Check if a joystick is connected. [More...](#)

static unsigned int **getButtonCount** (unsigned int joystick)
Return the number of buttons supported by a joystick.

static bool **hasAxis** (unsigned int joystick, **Axis** axis)
Check if a joystick supports a given axis. [More...](#)

static bool **isButtonPressed** (unsigned int joystick, unsigned int button)
Check if a joystick button is pressed. [More...](#)

static float **getAxisPosition** (unsigned int joystick, **Axis** axis)
Get the current position of a joystick axis. [More...](#)

static **Identification** **getIdentification** (unsigned int joystick)
Get the joystick information. [More...](#)

static void **update** ()
Update the states of all joysticks. [More...](#)

Detailed Description

Give access to the real-time state of the joysticks.

`sf::Joystick` provides an interface to the state of the joysticks.

It only contains static functions, so it's not meant to be instantiated. Instead, an index that is passed to the functions of this class.

This class allows users to query the state of joysticks at any time and directly from your window and its events. Compared to the JoystickMoved and JoystickButtonReleased events, `sf::Joystick` can retrieve the state of axes at any time (you don't need to store and update a boolean on your side in order to know if a button was released), and you always get the real state of joysticks, even if they are not connected or if your window is out of focus and no event is triggered.

SFML supports:

- 8 joysticks (`sf::Joystick::Count`)
- 32 buttons per joystick (`sf::Joystick::ButtonCount`)
- 8 axes per joystick (`sf::Joystick::AxisCount`)

Unlike the keyboard or mouse, the state of joysticks is sometimes not directly available (depending on the OS), therefore an `update()` function must be called in order to update the state. If you have a window with event handling, this is done automatically, you don't have to call `sf::Joystick::update`. If you have no window, or if you want to check joysticks state before using them, you have to call `sf::Joystick::update` explicitly.

Usage example:

```
// Is joystick #0 connected?
```

```
bool connected = sf::Joystick::isConnected(0);

// How many buttons does joystick #0 support?
unsigned int buttons = sf::Joystick::getButtonCount(0);

// Does joystick #0 define a X axis?
bool hasX = sf::Joystick::hasAxis(0, sf::Joystick::X);

// Is button #2 pressed on joystick #0?
bool pressed = sf::Joystick::isButtonPressed(0, 2);

// What's the current position of the Y axis on joystick #0?
float position = sf::Joystick::getAxisPosition(0, sf::Joystick::Y);
```

See also

[sf::Keyboard](#), [sf::Mouse](#)

Definition at line 41 of file [Joystick.hpp](#).

Member Enumeration Documentation

anonymous enum

Constants related to joysticks capabilities.

Enumerator	
Count	Maximum number of supported joysticks.
ButtonCount	Maximum number of supported buttons.
AxisCount	Maximum number of supported axes.

Definition at line 49 of file [Joystick.hpp](#).

enum sf::Joystick::Axis

Axes supported by SFML joysticks.

Enumerator	
X	The X axis.
Y	The Y axis.

Z	The Z axis.
R	The R axis.
U	The U axis.
V	The V axis.
PovX	The X axis of the point-of-view hat.
PovY	The Y axis of the point-of-view hat.

Definition at line 60 of file [Joystick.hpp](#).

Member Function Documentation

```
static float sf::Joystick::getAxisPosition ( unsigned int joystick,
                                             Axis      axis
                                           )
```

Get the current position of a joystick axis.

If the joystick is not connected, this function returns 0.

Parameters

joystick Index of the joystick
axis Axis to check

Returns

Current position of the axis, in range [-100 .. 100]

```
static unsigned int sf::Joystick::getButtonCount ( unsigned int joys
```

Return the number of buttons supported by a joystick.

If the joystick is not connected, this function returns 0.

Parameters

joystick Index of the joystick

Returns

Number of buttons supported by the joystick

```
static Identification sf::Joystick::getIdentification ( unsigned int joy )
```

Get the joystick information.

Parameters

joystick Index of the joystick

Returns

Structure containing joystick information.

```
static bool sf::Joystick::hasAxis ( unsigned int joystick,
                                            Axis      axis
                                  )
```

Check if a joystick supports a given axis.

If the joystick is not connected, this function returns false.

Parameters

joystick Index of the joystick

axis Axis to check

Returns

True if the joystick supports the axis, false otherwise

```
static bool sf::Joystick::isButtonPressed ( unsigned int joystick,
                                                    unsigned int button
                                         )
```

Check if a joystick button is pressed.

If the joystick is not connected, this function returns false.

Parameters

joystick Index of the joystick

button Button to check

Returns

True if the button is pressed, false otherwise

static bool sf::Joystick::isConnected (unsigned int joystick)

Check if a joystick is connected.

Parameters

joystick Index of the joystick to check

Returns

True if the joystick is connected, false otherwise

static void sf::Joystick::update ()

Update the states of all joysticks.

This function is used internally by SFML, so you normally don't have to call it. It may need to be called if you have no window yet (or no window at all): in this case, the joysticks will not be updated automatically.

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

- Joystick.hpp

SFML is licensed under the terms and conditions of the zlib/png license.
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Attributes](#) | [List of all members](#)

sf::Joystick::Identification Struct Reference

Structure holding a joystick's identification. More...

```
#include <Joystick.hpp>
```

Public Attributes

`String name`

Name of the joystick. [More...](#)

`unsigned int vendorId`

Manufacturer identifier. [More...](#)

`unsigned int productId`

Product identifier. [More...](#)

Detailed Description

Structure holding a joystick's identification.

Definition at line [76](#) of file [Joystick.hpp](#).

Member Data Documentation

String sf::Joystick::Identification::name

Name of the joystick.

Definition at line [80](#) of file [Joystick.hpp](#).

unsigned int sf::Joystick::Identification::productId

Product identifier.

Definition at line [82](#) of file [Joystick.hpp](#).

unsigned int sf::Joystick::Identification::vendorId

Manufacturer identifier.

Definition at line [81](#) of file [Joystick.hpp](#).

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

- [Joystick.hpp](#)
-

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Types](#) | [Static Public Member Functions](#) | [List of all members](#)

sf::Keyboard Class Reference

[Window module](#)

Give access to the real-time state of the keyboard. [More...](#)

#include <Keyboard.hpp>

Public Types

```
enum Key {  
    Unknown = -1, A = 0, B, C,  
    D, E, F, G,  
    H, I, J, K,  
    L, M, N, O,  
    P, Q, R, S,  
    T, U, V, W,  
    X, Y, Z, Num0,  
    Num1, Num2, Num3, Num4,  
    Num5, Num6, Num7, Num8,  
    Num9, Escape, LControl, LShift,  
    LAlt, LSystem, RControl, RShift,  
    RAlt, RSystem, Menu, LBracket,  
    RBracket, SemiColon, Comma, Period,  
    Quote, Slash, BackSlash, Tilde,  
    Equal, Dash, Space, Return,  
    BackSpace, Tab, PageUp, PageDown,  
    End, Home, Insert, Delete,  
    Add, Subtract, Multiply, Divide,  
    Left, Right, Up, Down,  
    Numpad0, Numpad1, Numpad2, Numpad3,  
    Numpad4, Numpad5, Numpad6, Numpad7,  
    Numpad8, Numpad9, F1, F2,  
    F3, F4, F5, F6,  
    F7, F8, F9, F10,  
    F11, F12, F13, F14,  
    F15, Pause, KeyCount  
}  
Key codes. More...
```

Static Public Member Functions

static bool `isKeyPressed (Key key)`

Check if a key is pressed. [More...](#)

static void `setVirtualKeyboardVisible (bool visible)`

Show or hide the virtual keyboard. [More...](#)

Detailed Description

Give access to the real-time state of the keyboard.

`sf::Keyboard` provides an interface to the state of the keyboard.

It only contains static functions (a single keyboard is assumed), so it's not

This class allows users to query the keyboard state at any time and dire window and its events. Compared to the KeyPressed and KeyReleased the state of a key at any time (you don't need to store and update a boole a key is pressed or released), and you always get the real state of the ke or released when your window is out of focus and no event is triggered.

Usage example:

```
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left))
{
    // move left...
}
else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right))
{
    // move right...
}
else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Escape))
{
    // quit...
}
```

See also

`sf::Joystick`, `sf::Mouse`, `sf::Touch`

Definition at line 40 of file `Keyboard.hpp`.

Member Enumeration Documentation

enum sf::Keyboard::Key

Key codes.

Enumerator	
Unknown	Unhandled key.
A	The A key.
B	The B key.
C	The C key.
D	The D key.
E	The E key.
F	The F key.
G	The G key.
H	

	The H key.
I	The I key.
J	The J key.
K	The K key.
L	The L key.
M	The M key.
N	The N key.
O	The O key.
P	The P key.
Q	The Q key.
R	The R key.
S	The S key.
T	The T key.
U	

	The U key.
V	The V key.
W	The W key.
X	The X key.
Y	The Y key.
Z	The Z key.
Num0	The 0 key.
Num1	The 1 key.
Num2	The 2 key.
Num3	The 3 key.
Num4	The 4 key.
Num5	The 5 key.
Num6	The 6 key.
Num7	

	The 7 key.
Num8	The 8 key.
Num9	The 9 key.
Escape	The Escape key.
LControl	The left Control key.
LShift	The left Shift key.
LAlt	The left Alt key.
LSystem	The left OS specific key: window (Windows and Linux), ap
RControl	The right Control key.
RShift	The right Shift key.
RAlt	The right Alt key.
RSystem	The right OS specific key: window (Windows and Linux), a
Menu	The Menu key.
LBracket	

	The [key.
RBracket	The] key.
SemiColon	The ; key.
Comma	The , key.
Period	The . key.
Quote	The ' key.
Slash	The / key.
BackSlash	The \ key.
Tilde	The ~ key.
Equal	The = key.
Dash	The - key.
Space	The Space key.
Return	The Return key.
BackSpace	

	The Backspace key.
Tab	The Tabulation key.
PageUp	The Page up key.
PageDown	The Page down key.
End	The End key.
Home	The Home key.
Insert	The Insert key.
Delete	The Delete key.
Add	The + key.
Subtract	The - key.
Multiply	The * key.
Divide	The / key.
Left	Left arrow.
Right	

	Right arrow.
Up	Up arrow.
Down	Down arrow.
Numpad0	The numpad 0 key.
Numpad1	The numpad 1 key.
Numpad2	The numpad 2 key.
Numpad3	The numpad 3 key.
Numpad4	The numpad 4 key.
Numpad5	The numpad 5 key.
Numpad6	The numpad 6 key.
Numpad7	The numpad 7 key.
Numpad8	The numpad 8 key.
Numpad9	The numpad 9 key.
F1	

	The F1 key.
F2	The F2 key.
F3	The F3 key.
F4	The F4 key.
F5	The F5 key.
F6	The F6 key.
F7	The F7 key.
F8	The F8 key.
F9	The F9 key.
F10	The F10 key.
F11	The F11 key.
F12	The F12 key.
F13	The F13 key.
F14	

	The F14 key.
F15	The F15 key.
Pause	The Pause key.
KeyCount	Keep last – the total number of keyboard keys.

Definition at line 48 of file [Keyboard.hpp](#).

Member Function Documentation

static bool sf::Keyboard::isKeyPressed (Key key)

Check if a key is pressed.

Parameters

key Key to check

Returns

True if the key is pressed, false otherwise

static void sf::Keyboard::setVirtualKeyboardVisible (bool visible)

Show or hide the virtual keyboard.

Warning: the virtual keyboard is not supported on all systems. It will typ OSes (Android, iOS) but not on desktop OSes (Windows, Linux, ...).

If the virtual keyboard is not available, this function does nothing.

Parameters

visible True to show, false to hide

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

- [Keyboard.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Types](#) | [Static Public Member Functions](#) | [List of all members](#)

sf::Mouse Class Reference

[Window module](#)

Give access to the real-time state of the mouse. More...

```
#include <Mouse.hpp>
```

Public Types

```
    Button {  
enum   Left, Right, Middle, XButton1,  
          XButton2, ButtonCount  
    }  
Mouse buttons. More...
```

```
enum Wheel { VerticalWheel, HorizontalWheel }  
Mouse wheels. More...
```

Static Public Member Functions

static bool `isButtonPressed (Button button)`

Check if a mouse button is pressed. [More...](#)

static `Vector2i getPosition ()`

Get the current position of the mouse in desktop coordinates.

static `Vector2i getPosition (const Window &relativeTo)`

Get the current position of the mouse in window coordinates.

static void `setPosition (const Vector2i &position)`

Set the current position of the mouse in desktop coordinates.

static void `setPosition (const Vector2i &position, const Window &relativeTo)`

Set the current position of the mouse in window coordinates.

Detailed Description

Give access to the real-time state of the mouse.

`sf::Mouse` provides an interface to the state of the mouse.

It only contains static functions (a single mouse is assumed), so it's not meant to be instantiated.

This class allows users to query the mouse state at any time and directly access the window and its events. Compared to the `MouseMoved`, `MouseButtonPressed` and `MouseButtonReleased` events, `sf::Mouse` can retrieve the state of the cursor and the buttons at any time and update a boolean on your side in order to know if a button is pressed or released without waiting for an event to be triggered. It also provides methods to get the current position of the mouse pointer in global coordinates or relative to a specific window.

The `setPosition` and `getPosition` functions can be used to change or retrieve the position of the mouse pointer. There are two versions: one that operates in global coordinates and one that operates in window coordinates (relative to a specific window).

Usage example:

```
if (sf::Mouse::isButtonPressed(sf::Mouse::Left))
{
    // left click...
}

// get global mouse position
sf::Vector2i position = sf::Mouse::getPosition();

// set mouse position relative to a window
sf::Mouse::setPosition(sf::Vector2i(100, 200), window);
```

See also

`sf::Joystick`, `sf::Keyboard`, `sf::Touch`

Definition at line 43 of file [Mouse.hpp](#).

Member Enumeration Documentation

enum sf::Mouse::Button

Mouse buttons.

Enumerator

Left	The left mouse button.
Right	The right mouse button.
Middle	The middle (wheel) mouse button.
XButton1	The first extra mouse button.
XButton2	The second extra mouse button.
ButtonCount	Keep last – the total number of mouse buttons.

Definition at line 51 of file [Mouse.hpp](#).

enum sf::Mouse::Wheel

Mouse wheels.

Enumerator	
VerticalWheel	The vertical mouse wheel.
HorizontalWheel	The horizontal mouse wheel.

Definition at line 66 of file [Mouse.hpp](#).

Member Function Documentation

static Vector2i sf::Mouse::getPosition()

Get the current position of the mouse in desktop coordinates.

This function returns the global position of the mouse cursor on the desk

Returns

Current position of the mouse

static Vector2i sf::Mouse::getPosition(const Window & relativeTo)

Get the current position of the mouse in window coordinates.

This function returns the current position of the mouse cursor, relative to

Parameters

relativeTo Reference window

Returns

Current position of the mouse

static bool sf::Mouse::isButtonPressed(Button button)

Check if a mouse button is pressed.

Parameters

button Button to check

Returns

True if the button is pressed, false otherwise

static void sf::Mouse::setPosition (const Vector2i & position)

Set the current position of the mouse in desktop coordinates.

This function sets the global position of the mouse cursor on the desktop

Parameters

position New position of the mouse

static void sf::Mouse::setPosition (const Vector2i & position, const Window & relativeTo)

Set the current position of the mouse in window coordinates.

This function sets the current position of the mouse cursor, relative to the

Parameters

position New position of the mouse

relativeTo Reference window

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

- [Mouse.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Types](#) | [Static Public Member Functions](#) | [List of all members](#)

sf::Sensor Class Reference

[Window module](#)

Give access to the real-time state of the sensors. [More...](#)

```
#include <Sensor.hpp>
```

Public Types

```
Type {  
enum Accelerometer, Gyroscope, Magnetometer, Gravity,  
      UserAcceleration, Orientation, Count  
}  
Sensor type. More...
```

Static Public Member Functions

static bool **isAvailable** (**Type** sensor)

Check if a sensor is available on the underlying platform.

static void **setEnabled** (**Type** sensor, bool enabled)

Enable or disable a sensor. [More...](#)

static **Vector3f** **getValue** (**Type** sensor)

Get the current sensor value. [More...](#)

Detailed Description

Give access to the real-time state of the sensors.

`sf::Sensor` provides an interface to the state of the various sensors that a

It only contains static functions, so it's not meant to be instantiated.

This class allows users to query the sensors values at any time and dire window and its events. Compared to the `SensorChanged` event, `sf::Se` sensor at any time (you don't need to store and update its current value or

Depending on the OS and hardware of the device (phone, tablet, ...), available. You should always check the availability of a sensor be `sf::Sensor::isAvailable` function.

You may wonder why some sensor types look so similar, for example `UserAcceleration`. The first one is the raw measurement of the acceleration of the earth gravity and the user movement. The others are more precise separately, which is usually more useful. In fact they are not direct sensors based on the raw acceleration and other sensors. This is exactly the same

Because sensors consume a non-negligible amount of current, they are called `sf::Sensor::setEnabled` for each sensor in which you are interested.

Usage example:

```
if (sf::Sensor::isAvailable(sf::Sensor::Gravity))
{
    // gravity sensor is available
}

// enable the gravity sensor
sf::Sensor::setEnabled(sf::Sensor::Gravity, true);
```

```
// get the current value of gravity
sf::Vector3f gravity = sf::Sensor::getValue(sf::Sensor::Gravity);
```

Definition at line 42 of file [Sensor.hpp](#).

Member Enumeration Documentation

enum sf::Sensor::Type

Sensor type.

Enumerator	
Accelerometer	Measures the raw acceleration (m/s ²)
Gyroscope	Measures the raw rotation rates (degrees/s)
Magnetometer	Measures the ambient magnetic field (micro-teslas)
Gravity	Measures the direction and intensity of gravity, independent of device orientation (m/s ²)
UserAcceleration	Measures the direction and intensity of device acceleration relative to gravity (m/s ²)
Orientation	Measures the absolute 3D orientation (degrees)
Count	Keep last – the total number of sensor types.

Definition at line 50 of file Sensor.hpp.

Member Function Documentation

static Vector3f sf::Sensor::getValue (Type sensor)

Get the current sensor value.

Parameters

sensor Sensor to read

Returns

The current sensor value

static bool sf::Sensor::isAvailable (Type sensor)

Check if a sensor is available on the underlying platform.

Parameters

sensor Sensor to check

Returns

True if the sensor is available, false otherwise

**static void sf::Sensor::setEnabled (Type sensor,
 bool enabled
)**

Enable or disable a sensor.

All sensors are disabled by default, to avoid consuming too much bandwidth. Once enabled, it starts sending events of the corresponding type.

This function does nothing if the sensor is unavailable.

Parameters

sensor Sensor to enable

enabled True to enable, false to disable

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

- [Sensor.hpp](#)

SFML is licensed under the terms and conditions of the [zlib/png license](#).

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#).

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Static Public Member Functions](#) | [List of all members](#)

sf::Touch Class Reference

[Window module](#)

Give access to the real-time state of the touches. [More...](#)

```
#include <Touch.hpp>
```

Static Public Member Functions

static bool `isDown` (`unsigned int finger`)

Check if a touch event is currently down. [More...](#)

static `Vector2i` `getPosition` (`unsigned int finger`)

Get the current position of a touch in desktop coordinates

static `Vector2i` `getPosition` (`unsigned int finger, const Window &relativeTo`)

Get the current position of a touch in window coordinates

Detailed Description

Give access to the real-time state of the touches.

`sf::Touch` provides an interface to the state of the touches.

It only contains static functions, so it's not meant to be instantiated.

This class allows users to query the touches state at any time and directly from the window and its events. Compared to the TouchBegan, TouchMoved and TouchEnded events, it retrieve the state of the touches at any time (you don't need to store and sort them in order to know if a touch is down), and you always get the real state of the touches when your window is out of focus and no event is triggered.

The getPosition function can be used to retrieve the current position of a touch that operates in global coordinates (relative to the desktop) and relative coordinates (relative to a specific window).

Touches are identified by an index (the "finger"), so that in multi-touch mode they are tracked correctly. As long as a finger touches the screen, it will keep the same index until start or stop touching the screen in the meantime. As a consequence, actions are sequential (i.e. touch number 0 may be released while touch number 1 is still down).

Usage example:

```
if (sf::Touch::isDown(0))
{
    // touch 0 is down
}

// get global position of touch 1
sf::Vector2i globalPos = sf::Touch::getPosition(1);

// get position of touch 1 relative to a window
sf::Vector2i relativePos = sf::Touch::getPosition(1, window);
```

See also

[sf::Joystick](#), [sf::Keyboard](#), [sf::Mouse](#)

Definition at line 43 of file [Touch.hpp](#).

Member Function Documentation

static Vector2i sf::Touch::getPosition (unsigned int finger)

Get the current position of a touch in desktop coordinates.

This function returns the current touch position in global (desktop) coordinates.

Parameters

finger Finger index

Returns

Current position of *finger*, or undefined if it's not down

**static Vector2i sf::Touch::getPosition (unsigned int finger,
const Window & relativeTo
)**

Get the current position of a touch in window coordinates.

This function returns the current touch position relative to the given window.

Parameters

finger Finger index

relativeTo Reference window

Returns

Current position of *finger*, or undefined if it's not down

static bool sf::Touch::isDown (unsigned int finger)

Check if a touch event is currently down.

Parameters

finger Finger index

Returns

True if *finger* is currently touching the screen, false otherwise

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

- [Touch.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

[Public Member Functions](#) | [Static Public Member Functions](#) | [Public Attributes](#) | [Related Functions](#) | [List of all members](#)

sf::VideoMode Class Reference

[Window module](#)

VideoMode defines a video mode (width, height, bpp) [More...](#)

```
#include <VideoMode.hpp>
```

Public Member Functions

`VideoMode ()`

Default constructor. [More...](#)

`VideoMode (unsigned int modeWidth, unsigned int modeHeight, ur`

Construct the video mode with its attributes. [More...](#)

`bool isValid () const`

Tell whether or not the video mode is valid. [More...](#)

Static Public Member Functions

```
static VideoMode getDesktopMode ()
```

Get the current desktop video mode. More...

```
static const std::vector<VideoMode> &
```

```
getFullscreenModes ()
```

Retrieve all the video modes supported in fullscr

Public Attributes

`unsigned int width`

Video mode width, in pixels. [More...](#)

`unsigned int height`

Video mode height, in pixels. [More...](#)

`unsigned int bitsPerPixel`

Video mode pixel depth, in bits per pixels. [More...](#)

Related Functions

(Note that these are not member functions.)

bool `operator==` (const `VideoMode` &left, const `VideoMode` &right)
Overload of == operator to compare two video modes. [More...](#)

bool `operator!=` (const `VideoMode` &left, const `VideoMode` &right)
Overload of != operator to compare two video modes. [More...](#)

bool `operator<` (const `VideoMode` &left, const `VideoMode` &right)
Overload of < operator to compare video modes. [More...](#)

bool `operator>` (const `VideoMode` &left, const `VideoMode` &right)
Overload of > operator to compare video modes. [More...](#)

bool `operator<=` (const `VideoMode` &left, const `VideoMode` &right)
Overload of <= operator to compare video modes. [More...](#)

bool `operator>=` (const `VideoMode` &left, const `VideoMode` &right)
Overload of >= operator to compare video modes. [More...](#)

Detailed Description

`VideoMode` defines a video mode (width, height, bpp)

A video mode is defined by a width and a height (in pixels) and a depth (in

Video modes are used to setup windows (`sf::Window`) at creation time.

The main usage of video modes is for fullscreen mode: indeed you must
allowed by the OS (which are defined by what the monitor and the graph
window creation will just fail).

`sf::VideoMode` provides a static function for retrieving the list of all the
system: `getFullscreenModes()`.

A custom video mode can also be checked directly for fullscreen compatibility:

Additionally, `sf::VideoMode` provides a static function to get the mode
`getDesktopMode()`. This allows to build windows with the same size or pix

Usage example:

```
// Display the list of all the video modes available for fullscreen
std::vector<sf::VideoMode> modes = sf::VideoMode::getFullscreenModes()
for (std::size_t i = 0; i < modes.size(); ++i)
{
    sf::VideoMode mode = modes[i];
    std::cout << "Mode #" << i << ": "
        << mode.width << "x" << mode.height << " - "
        << mode.bitsPerPixel << " bpp" << std::endl;
}

// Create a window with the same pixel depth as the desktop
sf::VideoMode desktop = sf::VideoMode::getDesktopMode();
window.create(sf::VideoMode(1024, 768, desktop.bitsPerPixel), "SFML window")
```

Definition at line 41 of file `VideoMode.hpp`.

Constructor & Destructor Documentation

sf::VideoMode::VideoMode ()

Default constructor.

This constructor initializes all members to 0.

```
sf::VideoMode::VideoMode ( unsigned int modeWidth,  
                           unsigned int modeHeight,  
                           unsigned int modeBitsPerPixel = 32  
                         )
```

Construct the video mode with its attributes.

Parameters

modeWidth Width in pixels

modeHeight Height in pixels

modeBitsPerPixel Pixel depths in bits per pixel

Member Function Documentation

static `VideoMode` `sf::VideoMode::getDesktopMode ()`

Get the current desktop video mode.

Returns

Current desktop video mode

static const std::vector<`VideoMode`>& `sf::VideoMode::getFullscreenModes ()`

Retrieve all the video modes supported in fullscreen mode.

When creating a fullscreen window, the video mode is restricted to be compatible with the video card driver and monitor support. This function returns the complete list of all supported video modes in fullscreen mode. The returned array is sorted from best to worst, so that the best mode (higher width, height and bits-per-pixel).

Returns

Array containing all the supported fullscreen modes

`bool sf::VideoMode::isValid () const`

Tell whether or not the video mode is valid.

The validity of video modes is only relevant when using fullscreen windows.

can be used with no restriction.

Returns

True if the video mode is valid for fullscreen mode

Friends And Related Function Documentation

```
bool operator!= ( const VideoMode & left,  
                  const VideoMode & right  
                )
```

Overload of != operator to compare two video modes.

Parameters

left Left operand (a video mode)
right Right operand (a video mode)

Returns

True if modes are different

```
bool operator< ( const VideoMode & left,  
                  const VideoMode & right  
                )
```

Overload of < operator to compare video modes.

Parameters

left Left operand (a video mode)
right Right operand (a video mode)

Returns

True if *left* is lesser than *right*

```
bool operator<= ( const VideoMode & left,  
                  const VideoMode & right  
                )
```

Overload of `<=` operator to compare video modes.

Parameters

left Left operand (a video mode)
right Right operand (a video mode)

Returns

True if *left* is lesser or equal than *right*

```
bool operator== ( const VideoMode & left,  
                  const VideoMode & right  
                )
```

Overload of `==` operator to compare two video modes.

Parameters

left Left operand (a video mode)
right Right operand (a video mode)

Returns

True if modes are equal

```
bool operator> ( const VideoMode & left,  
                  const VideoMode & right  
                )
```

Overload of `>` operator to compare video modes.

Parameters

- left** Left operand (a video mode)
- right** Right operand (a video mode)

Returns

True if *left* is greater than *right*

```
bool operator>= ( const VideoMode & left,  
                   const VideoMode & right  
                 )
```

Overload of `>=` operator to compare video modes.

Parameters

- left** Left operand (a video mode)
- right** Right operand (a video mode)

Returns

True if *left* is greater or equal than *right*

Member Data Documentation

unsigned int sf::VideoMode::bitsPerPixel

Video mode pixel depth, in bits per pixels.

Definition at line [104](#) of file `VideoMode.hpp`.

unsigned int sf::VideoMode::height

Video mode height, in pixels.

Definition at line [103](#) of file `VideoMode.hpp`.

unsigned int sf::VideoMode::width

Video mode width, in pixels.

Definition at line [102](#) of file `VideoMode.hpp`.

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

- [VideoMode.hpp](#)
-

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Public Member Functions](#) | [Protected Member Functions](#) | [Static Private Member Functions](#) | [List of all members](#)

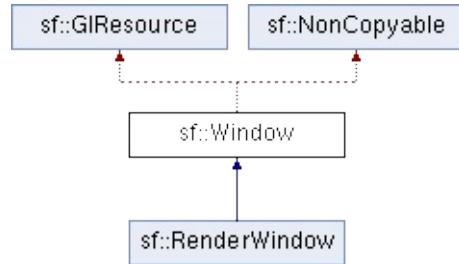
sf::Window Class Reference

Window module

Window that serves as a target for OpenGL rendering. More...

```
#include <Window.hpp>
```

Inheritance diagram for sf::Window:



Public Member Functions

`Window ()`

Default constructor. [More...](#)

`Window (VideoMode mode, const String &title, const ContextSettings &settings=ContextSettings())`
Construct a new window. [More...](#)

`Window (WindowHandle handle, const ContextSettings &settings=ContextSettings())`
Construct the window from an existing control.

`virtual ~Window ()`

Destructor. [More...](#)

`void create (VideoMode mode, const String &title, const ContextSettings &settings=ContextSettings())`
Create (or recreate) the window. [More...](#)

`void create (WindowHandle handle, const ContextSettings &settings=ContextSettings())`
Create (or recreate) the window from an existing control.

`void close ()`

Close the window and destroy all the attached contexts.

`bool isOpen () const`

Tell whether or not the window is open. [More...](#)

`const ContextSettings & getSettings () const`

Get the settings of the OpenGL context of the window.

`bool pollEvent (Event &event)`

Pop the event on top of the event queue, if any

`bool waitEvent (Event &event)`

Wait for an event and return it. [More...](#)

`Vector2i getPosition () const`

Get the position of the window. [More...](#)

`void setPosition (const Vector2i &position)`

Change the position of the window on screen. [More...](#)

`Vector2u getSize () const`

Get the size of the rendering region of the wind

`void setSize (const Vector2u &size)`

Change the size of the rendering region of the window. [More...](#)

`void setTitle (const String &title)`

Change the title of the window. [More...](#)

`void setIcon (unsigned int width, unsigned int height)`

Change the window's icon. [More...](#)

`void setVisible (bool visible)`

Show or hide the window. [More...](#)

`void setVerticalSyncEnabled (bool enabled)`

Enable or disable vertical synchronization. [More...](#)

`void setMouseCursorVisible (bool visible)`

Show or hide the mouse cursor. [More...](#)

`void setMouseCursorGrabbed (bool grabbed)`

Grab or release the mouse cursor. [More...](#)

`void setKeyRepeatEnabled (bool enabled)`
Enable or disable automatic key-repeat. [More..](#)

`void setFrameRateLimit (unsigned int limit)`
Limit the framerate to a maximum fixed frequency.

`void setJoystickThreshold (float threshold)`
Change the joystick threshold. [More...](#)

`bool setActive (bool active=true) const`
Activate or deactivate the window as the current active window.
[More...](#)

`void requestFocus ()`
Request the current window to be made the active window.

`bool hasFocus () const`
Check whether the window has the input focus.

`void display ()`
Display on screen what has been rendered to the window.

`WindowHandle getSystemHandle () const`
Get the OS-specific handle of the window. [More..](#)

Protected Member Functions

`virtual void onCreate ()`

Function called after the window has been created. [More...](#)

`virtual void onResize ()`

Function called after the window has been resized. [More...](#)

Static Private Member Functions

```
static void ensureGLContext ()
```

Empty function for ABI compatibility, use acquireTransientContext()

Detailed Description

Window that serves as a target for OpenGL rendering.

`sf::Window` is the main class of the `Window` module.

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

A `sf::Window` can create its own new window, or be embedded into an existing window using the `create(handle)` function. This can be useful for embedding an OpenGL rendering area into a window created by another (probably richer) GUI library.

The `sf::Window` class provides a simple interface for manipulating the window's position, control mouse cursor, etc. It also provides event handling through its `pollEvent()` method.

Note that OpenGL experts can pass their own parameters (antialiasing levels, depth and stencil buffers, etc.) to the OpenGL context attached to the window, with the `sf::ContextSettings` object passed as an optional argument when creating the window.

Usage example:

```
// Declare and create a new window
sf::Window window(sf::VideoMode(800, 600), "SFML window");

// Limit the framerate to 60 frames per second (this step is optional)
window.setFramerateLimit(60);

// The main loop - ends as soon as the window is closed
while (window.isOpen())
{
    // Event processing
    sf::Event event;
    while (window.pollEvent(event))
    {
        // Request for closing the window
        if (event.type == sf::Event::Closed)
```

```
        window.close();
    }

// Activate the window for OpenGL rendering
window.setActive();

// OpenGL drawing commands go here...

// End the current frame and display its contents on screen
window.display();
}
```

Definition at line 57 of file [Window/Window.hpp](#).

Constructor & Destructor Documentation

`sf::Window::Window()`

Default constructor.

This constructor doesn't actually create the window, use the other constr

```
sf::Window::Window( VideoMode mode,  
                    const String & title,  
                    Uint32 style = style::Default,  
                    const ContextSettings & settings = ContextSettings::Default)
```

Construct a new window.

This constructor creates the window with the size and pixel depth defined by `mode`. The `style` parameter can be passed to customize the look and behavior of the window (borders, title bar, etc.). If `style` contains `Style::Fullscreen`, then `mode` must be a valid video mode.

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

Parameters

- mode** Video mode to use (defines the width, height and depth of the window)
- title** Title of the window
- style** Window style, a bitwise OR combination of `sf::Style` enum values

settings Additional settings for the underlying OpenGL context

```
sf::Window::Window ( WindowHandle handle,  
                    const ContextSettings & settings = ContextSettings::Default  
)
```

Construct the window from an existing control.

Use this constructor if you want to create an OpenGL rendering area integrated with an existing windowing system control.

The second parameter is an optional structure specifying advanced OpenGL 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 ( )
```

Destructor.

Closes the window and frees 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` instance remains valid and you can still interact with it. All other functions such as `pollEvent()` or `display()` will still work (they will return `false` when `isOpen()` is called), and will have no effect on closed windows.

void sf::Window::create (VideoMode mode, const String & title, Uint32 style = Style::Default, const ContextSettings & settings = contextSettings())

Create (or recreate) the window.

If the window was already created, it closes it first. If `style` contains `Style::Close`, the window is closed without asking for confirmation.

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

Parameters

mode Video mode to use (defines the width, height and depth of the window)

title Title of the window

style Window style, a bitwise OR combination of sf::Style enum
settings Additional settings for the underlying OpenGL context

```
void sf::Window::create ( WindowHandle           handle,
                        const ContextSettings & settings = contextSettings )
```

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

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

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

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, to show it on screen.

```
Vector2i sf::Window::getPosition ( ) const
```

Get the position of the window.

Returns

Position of the window, in pixels

See also

[setPosition](#)

const ContextSettings& 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 function, if one or more settings were not supported. In this case, SFML creates a new structure containing the supported settings.

Returns

Structure containing the OpenGL context settings

Vector2u 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.

Returns

Size in pixels

See also

[setSize](#)

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 typed pointer to the OS. You shouldn't need to use this function, unless you have very specific needs. SFML doesn't support, or implement a temporary workaround until a bug is fixed.

Returns

System handle of the window

bool sf::Window::hasFocus () const

Check whether the window has the input focus.

At any given time, only one window may have the input focus to receive most keyboard and mouse events.

Returns

True if window has focus, false otherwise

See also

[requestFocus](#)

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 is still considered open (therefore this function would return true).

Returns

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

virtual void sf::Window::onCreate()

Function called after the window has been created.

This function is called so that derived classes can perform their own specific actions after the window is created.

Reimplemented in [sf::RenderWindow](#).

virtual void sf::Window::onResize()

Function called after the window has been resized.

This function is called so that derived classes can perform custom actions after the window has been resized.

Reimplemented in [sf::RenderWindow](#).

bool sf::Window::pollEvent(Event & 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 unmodified. Note that more than one event may be present in the event queue. You must call this function in a loop to make sure that you process every pending event.

```
sf::Event event;
while (window.pollEvent(event))
{
    // process event...
}
```

Parameters

event Event to be returned

Returns

True if an event was returned, or false if the event queue was empty

See also

[waitEvent](#)

void sf::Window::requestFocus()

Request the current window to be made the active foreground window.

At any given time, only one window may have the input focus to receive or mouse events. If a window requests focus, it only hints to the operating system. The operating system is free to deny the request. This is not to

See also

[hasFocus](#)

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, deactivate it on the previous thread first if it was active. Only one window can be active at a time, thus the window previously active (if any) automatically gets deactivated when this function is called with [requestFocus\(\)](#).

Parameters

active True to activate, false to deactivate

Returns

True 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()`. If a frame lasted long enough to match the framerate limit, SFML will try to catch up as much as it can, but since it internally uses `sf::sleep`, whose precision depends on the operating system, it may be a little unprecise as well (for example, you can get 65 FPS when setting a 60 FPS limit).

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. The pixels are copied from source alive after calling this function.

See also

[setTitle](#)

void sf::Window::setJoystickThreshold (float threshold)

Change the joystick threshold.

The joystick threshold is the value below which no JoystickMoved event is triggered.

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 whenever the key is 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::setMouseCursorGrabbed (bool grabbed)

Grab or release the mouse cursor.

If set, grabs the mouse cursor inside this window's client area so it may bounds. Note that grabbing is only active while the window has focus.

Parameters

grabbed 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 & position)

Change the position of the window on screen.

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

Parameters

position New position, in pixels

See also

[getPosition](#)

void sf::Window::setSize (const Vector2u & size)

Change the size of the rendering region of the window.

Parameters

size New size, in pixels

See also

[getSize](#)

void sf::Window::setTitle (const String & title)

Change the title of the window.

Parameters

title New title

See also

[setIcon](#)

void sf::Window::setVerticalSyncEnabled (bool enabled)

Enable or disable vertical synchronization.

Activating vertical synchronization will limit the number of frames displayed by the monitor. This can avoid some visual artifacts, and limit the framerate to the same value 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 & event)

Wait for an event and return it.

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

```
sf::Event event;
if (window.waitEvent(event))
{
    // process event...
}
```

Parameters

event Event to be returned

Returns

False if any error occurred

See also

[pollEvent](#)

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

- [Window/Window.hpp](#)
-

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

Namespaces

[Classes](#)

[Files](#)

[Namespace List](#)

[Namespace Members](#)

Namespace List

Here is a list of all documented namespaces with brief descriptions:

▼  sf

 **Gls** Namespace with GLSL types

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

Namespaces

[Classes](#)

[Files](#)

[Namespace List](#)

Namespace Members

[All](#)

[Typedefs](#)

Here is a list of all documented namespace members with links to the namespaces they belong to.

- [Bvec2 : sf::Gsl](#)
- [Bvec3 : sf::Gsl](#)
- [Bvec4 : sf::Gsl](#)
- [Ivec2 : sf::Gsl](#)
- [Ivec3 : sf::Gsl](#)
- [Ivec4 : sf::Gsl](#)
- [Mat3 : sf::Gsl](#)
- [Mat4 : sf::Gsl](#)
- [Vec2 : sf::Gsl](#)
- [Vec3 : sf::Gsl](#)
- [Vec4 : sf::Gsl](#)

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[**Namespaces**](#)[Classes](#)[Files](#)[Namespace List](#)[**Namespace Members**](#)[All](#)[**TypeDefs**](#)

- [Bvec2 : sf::Glsl](#)
- [Bvec3 : sf::Glsl](#)
- [Bvec4 : sf::Glsl](#)
- [Ivec2 : sf::Glsl](#)
- [Ivec3 : sf::Glsl](#)
- [Ivec4 : sf::Glsl](#)
- [Mat3 : sf::Glsl](#)
- [Mat4 : sf::Glsl](#)
- [Vec2 : sf::Glsl](#)
- [Vec3 : sf::Glsl](#)
- [Vec4 : sf::Glsl](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

Class List

Here are the classes, structs, unions and interfaces with brief descriptions

▼ N sf

 AIResource	Base class for classes that require an C
 BlendMode	Blending modes for drawing
 CircleShape	Specialized shape representing a circle
 Clock	Utility class that measures the elapsed
 Color	Utility class for manipulating RGBA colc
 Context	Class holding a valid drawing context
 ContextSettings	Structure defining the settings of the Op
 ConvexShape	Specialized shape representing a conv
 Drawable	Abstract base class for objects that can
 Event	Defines a system event and its paramet
 JoystickButtonEvent	Joystick buttons events parameters (Jo
 JoystickConnectEvent	Joystick connection events parameters

		JoystickDisconnected)
⌚	JoystickMoveEvent	Joystick axis move event parameters (JoystickMoveEvent)
⌚	KeyEvent	Keyboard event parameters (KeyPress)
⌚	MouseButtonEvent	Mouse buttons events parameters (MouseButtonReleased)
⌚	MouseMoveEvent	Mouse move event parameters (MouseMove)
⌚	MouseWheelEvent	Mouse wheel events parameters (MouseWheel)
⌚	MouseWheelScrollView	Mouse wheel events parameters (MouseWheelScrollView)
⌚	SensorEvent	Sensor event parameters (SensorChangeEvent)
⌚	SizeEvent	Size events parameters (Resized)
⌚	TextEvent	Text event parameters (TextEntered)
⌚	TouchEvent	Touch events parameters (TouchBegan)
⌚	FileInputStream	Implementation of input stream based on file
▼ ⌚	Font	Class for loading and manipulating character fonts
⌚	Info	Holds various information about a font
▼ ⌚	Ftp	A FTP client
⌚	DirectoryResponse	Specialization of FTP response returning directory listing
⌚	ListingResponse	Specialization of FTP response returning a file listing
⌚	Response	Define a FTP response
▼ ⌚	GIResource	Base class for classes that require an OpenGL context
⌚	TransientContextLock	RAII helper class to temporarily lock an OpenGL context
⌚	Glyph	Structure describing a glyph
▼ ⌚	Http	A HTTP client
⌚	Request	Define a HTTP request
⌚	Response	Define a HTTP response
⌚	Image	Class for loading, manipulating and saving images

⌚ InputSoundFile	Provide read access to sound files
⌚ InputStream	Abstract class for custom file input streams
⌚ IpAddress	Encapsulate an IPv4 network address
▼⌚ Joystick	Give access to the real-time state of the joystick
⌚ Identification	Structure holding a joystick's identification information
⌚ Keyboard	Give access to the real-time state of the keyboard
⌚ Listener	The audio listener is the point in the scene from which sounds are heard
⌚ Lock	Automatic wrapper for locking and unlocking mutexes
⌚ MemoryInputStream	Implementation of input stream based on memory
⌚ Mouse	Give access to the real-time state of the mouse
⌚ Music	Streamed music played from an audio file
⌚ Mutex	Blocks concurrent access to shared resources
⌚ NonCopyable	Utility class that makes any derived class non-copyable
⌚ OutputSoundFile	Provide write access to sound files
⌚ Packet	Utility class to build blocks of data to transfer
⌚ Rect	Utility class for manipulating 2D axis aligned rectangles
⌚ RectangleShape	Specialized shape representing a rectangle
⌚ RenderStates	Define the states used for drawing to a render target
⌚ RenderTarget	Base class for all render targets (windows, textures)
⌚ RenderTexture	Target for off-screen 2D rendering into a texture
⌚ RenderWindow	Window that can serve as a target for 2D rendering
⌚ Sensor	Give access to the real-time state of the sensor
▼⌚ Shader	Shader class (vertex, geometry and fragment)
⌚ CurrentTextureType	Special type that can be passed to setTexture to indicate the texture of the object being drawn

⌚ Shape	Base class for textured shapes with outlines.
⌚ Socket	Base class for all the socket types.
⌚ SocketSelector	Multiplexer that allows to read from multiple sockets.
⌚ Sound	Regular sound that can be played in the background.
⌚ SoundBuffer	Storage for audio samples defining a sound's properties.
⌚ SoundBufferRecorder	Specialized SoundRecorder which stores audio samples in a SoundBuffer .
⌚ SoundFileFactory	Manages and instantiates sound file readers.
▼ ⌚ SoundFileReader	Abstract base class for sound file decoders.
⌚ Info	Structure holding the audio properties of a sound file.
⌚ SoundFileWriter	Abstract base class for sound file encoders.
⌚ SoundRecorder	Abstract base class for capturing sound.
⌚ SoundSource	Base class defining a sound's properties.
▼ ⌚ SoundStream	Abstract base class for streamed audio.
⌚ Chunk	Structure defining a chunk of audio data.
⌚ Sprite	Drawable representation of a texture, with color, etc.
⌚ String	Utility string class that automatically handles encodings.
⌚ TcpListener	Socket that listens to new TCP connections.
⌚ TcpSocket	Specialized socket using the TCP protocol.
⌚ Text	Graphical text that can be drawn to a render target.
⌚ Texture	Image living on the graphics card that can be drawn.
⌚ Thread	Utility class to manipulate threads.
⌚ ThreadLocal	Defines variables with thread-local storage.
⌚ ThreadLocalPtr	Pointer to a thread-local variable.
⌚ Time	Represents a time value.

C Touch	Give access to the real-time state of the touch screen.
C Transform	Define a 3x3 transform matrix
C Transformable	Decomposed transform defined by a position and orientation.
C UdpSocket	Specialized socket using the UDP protocol.
C Utf	Utility class providing generic functions
C Utf< 16 >	Specialization of the Utf template for UTF-16 strings.
C Utf< 32 >	Specialization of the Utf template for UTF-32 strings.
C Utf< 8 >	Specialization of the Utf template for UTF-8 strings.
C Vector2	Utility template class for manipulating 2D vectors.
C Vector3	Utility template class for manipulating 3D vectors.
C Vertex	Define a point with color and texture coordinates.
C VertexArray	Define a set of one or more 2D primitive vertices.
C VideoMode	VideoMode defines a video mode (width, height, resolution).
C View	2D camera that defines what region is shown on the screen.
C Window	Window that serves as a target for OpenGL rendering.

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Static Public Member Functions](#) | [List of all members](#)

sf::Utf< 16 > Class Template Reference

Specialization of the [Utf](#) template for UTF-16. [More...](#)

#include <Utf.hpp>

Static Public Member Functions

template<typename In >
 static In **decode** (In begin, In end, Uint32 &output, Uint32 replacement=0)
 Decode a single UTF-16 character. [More...](#)

template<typename Out >
 static Out **encode** (Uint32 input, Out output, Uint16 replacement=0)
 Encode a single UTF-16 character. [More...](#)

template<typename In >
 static In **next** (In begin, In end)
 Advance to the next UTF-16 character. [More...](#)

template<typename In >
 static std::size_t **count** (In begin, In end)
 Count the number of characters of a UTF-16 sequence

template<typename In , typename Out >
 static Out **fromAnsi** (In begin, In end, Out output, const std::locale &locale=std::locale())
 Convert an ANSI characters range to UTF-16. [More...](#)

template<typename In , typename Out >
 static Out **fromWide** (In begin, In end, Out output)
 Convert a wide characters range to UTF-16. [More...](#)

template<typename In , typename Out >
 static Out **fromLatin1** (In begin, In end, Out output)
 Convert a latin-1 (ISO-5589-1) characters range to UTF-16.

template<typename In , typename Out >
 static Out **toAnsi** (In begin, In end, Out output, char replacement=0, const std::locale &locale=std::locale())
 Convert an UTF-16 characters range to ANSI characters.

```
template<typename In , typename Out >
    static Out toWide (In begin, In end, Out output, wchar_t replacement)
        Convert an UTF-16 characters range to wide characters
```

```
template<typename In , typename Out >
    static Out toLatin1 (In begin, In end, Out output, char replacement)
        Convert an UTF-16 characters range to latin-1 (ISO-559)
```

```
template<typename In , typename Out >
    static Out toUtf8 (In begin, In end, Out output)
        Convert a UTF-16 characters range to UTF-8. More...
```

```
template<typename In , typename Out >
    static Out toUtf16 (In begin, In end, Out output)
        Convert a UTF-16 characters range to UTF-16. More...
```

```
template<typename In , typename Out >
    static Out toUtf32 (In begin, In end, Out output)
        Convert a UTF-16 characters range to UTF-32. More...
```

Detailed Description

```
template<>
class sf::Utf< 16 >
```

Specialization of the [Utf](#) template for UTF-16.

Definition at line 255 of file [Utf.hpp](#).

Member Function Documentation

```
template<typename In >
static std::size_t sf::Utf< 16 >::count ( In begin,
                                     In end
)
```

Count the number of characters of a UTF-16 sequence.

This function is necessary for multi-elements encodings, as a single character can be stored in multiple storage elements, thus the total size can be different from (begin - end).

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence

Returns

Iterator pointing to one past the last read element of the input sequence

```
template<typename In >
static In sf::Utf< 16 >::decode ( In      begin,
                                    In      end,
                                    Uint32 & output,
                                    Uint32 replacement = 0
)
```

Decode a single UTF-16 character.

Decoding a character means finding its unique 32-bits code (called standard).

Parameters

begin	Iterator pointing to the beginning of the input sequence
end	Iterator pointing to the end of the input sequence
output	Codepoint of the decoded UTF-16 character
replacement	Replacement character to use in case the UTF-8 sequence is invalid

Returns

Iterator pointing to one past the last read element of the input sequence.

```
template<typename Out >
static Out sf::Utf< 16 >::encode ( UInt32 input,
                           Out    output,
                           UInt16 replacement = 0
)
```

Encode a single UTF-16 character.

Encoding a character means converting a unique 32-bits code (called standard) into its representation in memory, called encoding, UTF-16.

Parameters

input	Codepoint to encode as UTF-16
output	Iterator pointing to the beginning of the output sequence
replacement	Replacement for characters not convertible to UTF-16

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 16 >::fromAnsi ( In begin,
                                         In end,
                                         Out output,
                                         const std::locale & locale = std::locale())

```

Convert an ANSI characters range to UTF-16.

The current global locale will be used by default, unless you pass a custom one.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence
locale Locale to use for conversion

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 16 >::fromLatin1 ( In begin,
                                         In end,
                                         Out output
                                         )

```

Convert a latin-1 (ISO-5589-1) characters range to UTF-16.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 16 >::fromWide ( In begin,
                                         In end,
                                         Out output
                                         )
```

Convert a wide characters range to UTF-16.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In >
static In sf::Utf< 16 >::next ( In begin,
                                    In end
                                    )
```

Advance to the next UTF-16 character.

This function is necessary for multi-elements encodings, as a single character is stored in two storage elements.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence

Returns

Iterator pointing to one past the last read element of the input sequence

```
template<typename In , typename Out >
static Out sf::Utf< 16 >::toAnsi ( In begin,
                                In end,
                                Out output,
                                char replacement = e
                                const std::locale & locale = std::loc
                                )
```

Convert an UTF-16 characters range to ANSI characters.

The current global locale will be used by default, unless you pass a custom one.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence
replacement Replacement for characters not convertible to ANSI (by default, '\0')
locale Locale to use for conversion

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 16 >::toLatin1 ( In begin,
                                    In end,
```

```
    Out output,  
    char replacement = 0  
)
```

Convert an UTF-16 characters range to latin-1 (ISO-5589-1) characters.

Parameters

begin	Iterator pointing to the beginning of the input sequence
end	Iterator pointing to the end of the input sequence
output	Iterator pointing to the beginning of the output sequence
replacement	Replacement for characters not convertible to wide (u8)

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >  
static Out sf::Utf< 16 >::toUtf16 ( In begin,  
                                    In end,  
                                    Out output  
)
```

Convert a UTF-16 characters range to UTF-16.

This functions does nothing more than a direct copy; it is defined only to support other specializations of the sf::Utf<> template, and allow generic code to

Parameters

begin	Iterator pointing to the beginning of the input sequence
end	Iterator pointing to the end of the input sequence
output	Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 16 >::toUtf32 ( In begin,
                                In end,
                                Out output
)
```

Convert a UTF-16 characters range to UTF-32.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 16 >::toUtf8 ( In begin,
                                In end,
                                Out output
)
```

Convert a UTF-16 characters range to UTF-8.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
```

```
static Out sf::Utf< 16 >::toWide( In      begin,
                                    In      end,
                                    Out     output,
                                    wchar_t replacement = 0
                                )
```

Convert an UTF-16 characters range to wide characters.

Parameters

begin	Iterator pointing to the beginning of the input sequence
end	Iterator pointing to the end of the input sequence
output	Iterator pointing to the beginning of the output sequence
replacement	Replacement for characters not convertible to wide (u)

Returns

Iterator to the end of the output sequence which has been written

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

- [Utf.hpp](#)

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Static Public Member Functions](#) | [List of all members](#)

sf::Utf< 32 > Class Template Reference

Specialization of the [Utf](#) template for UTF-32. [More...](#)

#include <Utf.hpp>

Static Public Member Functions

template<typename In >
 static In **decode** (In begin, In end, Uint32 &output, Uint32 replacement=0)
 Decode a single UTF-32 character. [More...](#)

template<typename Out >
 static Out **encode** (Uint32 input, Out output, Uint32 replacement=0)
 Encode a single UTF-32 character. [More...](#)

template<typename In >
 static In **next** (In begin, In end)
 Advance to the next UTF-32 character. [More...](#)

template<typename In >
 static std::size_t **count** (In begin, In end)
 Count the number of characters of a UTF-32 sequence

template<typename In , typename Out >
 static Out **fromAnsi** (In begin, In end, Out output, const std::locale &locale=std::locale())
 Convert an ANSI characters range to UTF-32. [More...](#)

template<typename In , typename Out >
 static Out **fromWide** (In begin, In end, Out output)
 Convert a wide characters range to UTF-32. [More...](#)

template<typename In , typename Out >
 static Out **fromLatin1** (In begin, In end, Out output)
 Convert a latin-1 (ISO-5589-1) characters range to UTF-32. [More...](#)

template<typename In , typename Out >
 static Out **toAnsi** (In begin, In end, Out output, char replacement='?')
 &locale=std::locale())
 Convert an UTF-32 characters range to ANSI characters.

```
template<typename In , typename Out >
    static Out toWide (In begin, In end, Out output, wchar_t replacement)
        Convert an UTF-32 characters range to wide characters.
```

```
template<typename In , typename Out >
    static Out toLatin1 (In begin, In end, Out output, char replacement)
        Convert an UTF-16 characters range to latin-1 (ISO-559).
```

```
template<typename In , typename Out >
    static Out toUtf8 (In begin, In end, Out output)
        Convert a UTF-32 characters range to UTF-8. More...
```

```
template<typename In , typename Out >
    static Out toUtf16 (In begin, In end, Out output)
        Convert a UTF-32 characters range to UTF-16. More...
```

```
template<typename In , typename Out >
    static Out toUtf32 (In begin, In end, Out output)
        Convert a UTF-32 characters range to UTF-32. More...
```

```
template<typename In >
    static Uint32 decodeAnsi (In input, const std::locale &locale=std::locale())
        Decode a single ANSI character to UTF-32. More...
```

```
template<typename In >
    static Uint32 decodeWide (In input)
        Decode a single wide character to UTF-32. More...
```

```
template<typename Out >
    static Out encodeAnsi (Uint32 codepoint, Out output, char replacement,
                           &locale=std::locale())
        Encode a single UTF-32 character to ANSI. More...
```

```
template<typename Out >
    static Out encodeWide (Uint32 codepoint, Out output, wchar_t replacement)
        Encode a single UTF-32 character to wide. More...
```

Detailed Description

```
template<>
class sf::Utf< 32 >
```

Specialization of the [Utf](#) template for UTF-32.

Definition at line [462](#) of file [Utf.hpp](#).

Member Function Documentation

```
template<typename In >
static std::size_t sf::Utf< 32 >::count ( In begin,
                                     In end
)
```

Count the number of characters of a UTF-32 sequence.

This function is trivial for UTF-32, which can store every character in a single 32-bit integer.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence

Returns

Iterator pointing to one past the last read element of the input sequence.

```
template<typename In >
static In sf::Utf< 32 >::decode ( In      begin,
                                    In      end,
                                    Uint32 & output,
                                    Uint32 replacement = 0
)
```

Decode a single UTF-32 character.

Decoding a character means finding its unique 32-bits code (called

standard. For UTF-32, the character value is the same as the codepoint.

Parameters

begin	Iterator pointing to the beginning of the input sequence
end	Iterator pointing to the end of the input sequence
output	Codepoint of the decoded UTF-32 character
replacement	Replacement character to use in case the UTF-8 sequence is invalid

Returns

Iterator pointing to one past the last read element of the input sequence.

```
template<typename In >
static Uint32 sf::Utf< 32 >::decodeAnsi ( In input,
                                         const std::locale & locale =
                                         )
```

Decode a single ANSI character to UTF-32.

This function does not exist in other specializations of `sf::Utf<>`, it is defined by several other conversion functions).

Parameters

input	Input ANSI character
locale	Locale to use for conversion

Returns

Converted character

```
template<typename In >
static Uint32 sf::Utf< 32 >::decodeWide ( In input )
```

Decode a single wide character to UTF-32.

This function does not exist in other specializations of `sf::Utf<>`, it is defined by several other conversion functions).

Parameters

input Input wide character

Returns

Converted character

```
template<typename Out >
static Out sf::Utf< 32 >::encode( UInt32 input,
                                    Out     output,
                                    UInt32 replacement = 0
                                )
```

Encode a single UTF-32 character.

Encoding a character means converting a unique 32-bits code (called encoding, UTF-32. For UTF-32, the codepoint is the same as the character).

Parameters

input Codepoint to encode as UTF-32

output Iterator pointing to the beginning of the output sequence

replacement Replacement for characters not convertible to UTF-32

Returns

Iterator to the end of the output sequence which has been written

```
template<typename Out >
static Out sf::Utf< 32 >::encodeAnsi ( UInt32 codepoint,
                                         Out output,
                                         char char,
                                         const std::locale & locale = std::locale(),
                                         wchar_t replacement = 0
                                         )
```

Encode a single UTF-32 character to ANSI.

This function does not exist in other specializations of `sf::Utf<>`, it is defined by several other conversion functions).

Parameters

codepoint	Iterator pointing to the beginning of the input sequence
output	Iterator pointing to the beginning of the output sequence
replacement	Replacement if the input character is not convertible to the output type
locale	Locale to use for conversion

Returns

Iterator to the end of the output sequence which has been written

```
template<typename Out >
static Out sf::Utf< 32 >::encodeWide ( UInt32 codepoint,
                                         Out output,
                                         wchar_t replacement = 0
                                         )
```

Encode a single UTF-32 character to wide.

This function does not exist in other specializations of `sf::Utf<>`, it is defined by several other conversion functions).

Parameters

codepoint Iterator pointing to the beginning of the input sequence
output Iterator pointing to the beginning of the output sequence
replacement Replacement if the input character is not convertible to the output character

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 32 >::fromAnsi ( In begin,
                                         In end,
                                         Out output,
                                         const std::locale & locale = std::locale())

```

Convert an ANSI characters range to UTF-32.

The current global locale will be used by default, unless you pass a custom one.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence
locale Locale to use for conversion

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 32 >::fromLatin1 ( In begin,
                                         In end,
                                         Out output)

```

```
Out output  
)
```

Convert a latin-1 (ISO-5589-1) characters range to UTF-32.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >  
static Out sf::Utf< 32 >::fromWide ( In begin,  
                                     In end,  
                                     Out output  
)
```

Convert a wide characters range to UTF-32.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In >  
static In sf::Utf< 32 >::next ( In begin,
```

```
In end  
)
```

Advance to the next UTF-32 character.

This function is trivial for UTF-32, which can store every character in a single 32-bit integer.

Parameters

- begin** Iterator pointing to the beginning of the input sequence
- end** Iterator pointing to the end of the input sequence

Returns

Iterator pointing to one past the last read element of the input sequence.

```
template<typename In , typename Out >  
static Out sf::Utf< 32 >::toAnsi( In begin,  
                                    In end,  
                                    Out output,  
                                    char replacement = e  
                                    const std::locale & locale = std::loc  
                                    )
```

Convert an UTF-32 characters range to ANSI characters.

The current global locale will be used by default, unless you pass a custom one.

Parameters

- begin** Iterator pointing to the beginning of the input sequence
- end** Iterator pointing to the end of the input sequence
- output** Iterator pointing to the beginning of the output sequence
- replacement** Replacement for characters not convertible to ANSI (by default, '\ufffd')
- locale** Locale to use for conversion

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 32 >::toLatin1 ( In begin,
                                     In end,
                                     Out output,
                                     char replacement = 0
)
```

Convert an UTF-16 characters range to latin-1 (ISO-5589-1) characters.

Parameters

begin	Iterator pointing to the beginning of the input sequence
end	Iterator pointing to the end of the input sequence
output	Iterator pointing to the beginning of the output sequence
replacement	Replacement for characters not convertible to wide (u)

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 32 >::toUtf16 ( In begin,
                                     In end,
                                     Out output
)
```

Convert a UTF-32 characters range to UTF-16.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 32 >::toUtf32 ( In   begin,
                                         In   end,
                                         Out  output
                                         )
```

Convert a UTF-32 characters range to UTF-32.

This functions does nothing more than a direct copy; it is defined only 1 other specializations of the sf::Utf<> template, and allow generic code to

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 32 >::toUtf8 ( In   begin,
                                         In   end,
                                         Out  output
                                         )
```

Convert a UTF-32 characters range to UTF-8.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 32 >::toWide( In      begin,
                                    In      end,
                                    Out     output,
                                    wchar_t replacement = 0
                                )
```

Convert an UTF-32 characters range to wide characters.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence
replacement Replacement for characters not convertible to wide (u)

Returns

Iterator to the end of the output sequence which has been written

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

- [Utf.hpp](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

[Static Public Member Functions](#) | [List of all members](#)

sf::Utf< 8 > Class Template Reference

Specialization of the [Utf](#) template for UTF-8. [More...](#)

#include <Utf.hpp>

Static Public Member Functions

template<typename In >
 static In **decode** (In begin, In end, Uint32 &output, Uint32 replace)
 Decode a single UTF-8 character. [More...](#)

template<typename Out >
 static Out **encode** (Uint32 input, Out output, Uint8 replacement=0)
 Encode a single UTF-8 character. [More...](#)

template<typename In >
 static In **next** (In begin, In end)
 Advance to the next UTF-8 character. [More...](#)

template<typename In >
 static std::size_t **count** (In begin, In end)
 Count the number of characters of a UTF-8 sequence.

template<typename In , typename Out >
 static Out **fromAnsi** (In begin, In end, Out output, const std::locale &locale)
 Convert an ANSI characters range to UTF-8. [More...](#)

template<typename In , typename Out >
 static Out **fromWide** (In begin, In end, Out output)
 Convert a wide characters range to UTF-8. [More...](#)

template<typename In , typename Out >
 static Out **fromLatin1** (In begin, In end, Out output)
 Convert a latin-1 (ISO-5589-1) characters range to UTF

template<typename In , typename Out >
 static Out **toAnsi** (In begin, In end, Out output, char replacement='?')
 Convert an UTF-8 characters range to ANSI characters

```
template<typename In , typename Out >
    static Out toWide (In begin, In end, Out output, wchar_t replacement)
        Convert an UTF-8 characters range to wide characters.
```

```
template<typename In , typename Out >
    static Out toLatin1 (In begin, In end, Out output, char replacement)
        Convert an UTF-8 characters range to latin-1 (ISO-5589).
```

```
template<typename In , typename Out >
    static Out toUtf8 (In begin, In end, Out output)
        Convert a UTF-8 characters range to UTF-8. More...
```

```
template<typename In , typename Out >
    static Out toUtf16 (In begin, In end, Out output)
        Convert a UTF-8 characters range to UTF-16. More...
```

```
template<typename In , typename Out >
    static Out toUtf32 (In begin, In end, Out output)
        Convert a UTF-8 characters range to UTF-32. More...
```

Detailed Description

```
template<>
class sf::Utf< 8 >
```

Specialization of the [Utf](#) template for UTF-8.

Definition at line [48](#) of file [Utf.hpp](#).

Member Function Documentation

```
template<typename In >
static std::size_t sf::Utf< 8 >::count ( In begin,
                                     In end
)
```

Count the number of characters of a UTF-8 sequence.

This function is necessary for multi-elements encodings, as a single character can be stored in multiple storage elements, thus the total size can be different from (begin - end).

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence

Returns

Iterator pointing to one past the last read element of the input sequence

```
template<typename In >
static In sf::Utf< 8 >::decode ( In      begin,
                                 In      end,
                                 Uint32 & output,
                                 Uint32   replacement = 0
)
```

Decode a single UTF-8 character.

Decoding a character means finding its unique 32-bits code (called standard).

Parameters

begin	Iterator pointing to the beginning of the input sequence
end	Iterator pointing to the end of the input sequence
output	Codepoint of the decoded UTF-8 character
replacement	Replacement character to use in case the UTF-8 sequence is invalid

Returns

Iterator pointing to one past the last read element of the input sequence.

```
template<typename Out >
static Out sf::Utf< 8 >::encode ( UInt32 input,
                           Out    output,
                           UInt8 replacement = 0
)
```

Encode a single UTF-8 character.

Encoding a character means converting a unique 32-bits code (called standard) into a sequence of bytes (called encoding), UTF-8.

Parameters

input	Codepoint to encode as UTF-8
output	Iterator pointing to the beginning of the output sequence
replacement	Replacement for characters not convertible to UTF-8 (e.g. surrogates)

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 8 >::fromAnsi ( In begin,
                                         In end,
                                         Out output,
                                         const std::locale & locale = std::locale())
                                         )
```

Convert an ANSI characters range to UTF-8.

The current global locale will be used by default, unless you pass a custom one.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence
locale Locale to use for conversion

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 8 >::fromLatin1 ( In begin,
                                         In end,
                                         Out output
                                         )
```

Convert a latin-1 (ISO-5589-1) characters range to UTF-8.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 8 >::fromWide ( In begin,
                                         In end,
                                         Out output
                                       )
```

Convert a wide characters range to UTF-8.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In >
static In sf::Utf< 8 >::next ( In begin,
                                    In end
                                  )
```

Advance to the next UTF-8 character.

This function is necessary for multi-elements encodings, as a single character is stored in multiple storage elements.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence

Returns

Iterator pointing to one past the last read element of the input sequence

```
template<typename In , typename Out >
static Out sf::Utf< 8 >::toAnsi ( In begin,
                                In end,
                                Out output,
                                char replacement = 0,
                                const std::locale & locale = std::locale( )
```

Convert an UTF-8 characters range to ANSI characters.

The current global locale will be used by default, unless you pass a custom one.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence
replacement Replacement for characters not convertible to ANSI (by default, a question mark)
locale Locale to use for conversion

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 8 >::toLatin1 ( In begin,
                                    In end,
```

```
    Out output,  
    char replacement = 0  
)
```

Convert an UTF-8 characters range to latin-1 (ISO-5589-1) characters.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence
replacement Replacement for characters not convertible to wide (u)

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >  
static Out sf::Utf< 8 >::toUtf16 ( In begin,  
                                    In end,  
                                    Out output  
)
```

Convert a UTF-8 characters range to UTF-16.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 8 >::toUtf32 ( In   begin,
                                         In   end,
                                         Out  output
                                         )
```

Convert a UTF-8 characters range to UTF-32.

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 8 >::toUtf8 ( In   begin,
                                         In   end,
                                         Out  output
                                         )
```

Convert a UTF-8 characters range to UTF-8.

This functions does nothing more than a direct copy; it is defined only to other specializations of the sf::Utf<> template, and allow generic code to

Parameters

begin Iterator pointing to the beginning of the input sequence
end Iterator pointing to the end of the input sequence
output Iterator pointing to the beginning of the output sequence

Returns

Iterator to the end of the output sequence which has been written

```
template<typename In , typename Out >
static Out sf::Utf< 8 >::toWide ( In      begin,
                                         In      end,
                                         Out     output,
                                         wchar_t replacement = 0
                                         )
```

Convert an UTF-8 characters range to wide characters.

Parameters

begin	Iterator pointing to the beginning of the input sequence
end	Iterator pointing to the end of the input sequence
output	Iterator pointing to the beginning of the output sequence
replacement	Replacement for characters not convertible to wide (u)

Returns

Iterator to the end of the output sequence which has been written

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

- [Utf.hpp](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

Class Index

[A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [H](#) | [I](#) | [J](#) | [K](#) | [L](#) | [M](#) | [N](#) | [O](#) | [P](#) | [F](#)

Glyph (sf)

A

AIResource (sf)

B

BlendMode (sf)

C

SoundStream::Chunk (sf)

CircleShape (sf)

Clock (sf)

Color (sf)

Context (sf)

ContextSettings (sf)

ConvexShape (sf)

Shader::CurrentTextureType (sf)

H

Http (sf)

E

I

Joystick::Identification (sf)

Image (sf)

SoundFileReader::Info (sf)

Font::Info (sf)

InputSoundFile (sf)

InputStream (sf)

IpAddress (sf)

J

Joystick (sf)

F

Event::JoystickButtonEvent (sf)

D

Ftp::DirectoryResponse (sf)
Drawable (sf)

E

Event (sf)

F

FileInputStream (sf)
Font (sf)
Ftp (sf)

G

GIResource (sf)

Event::JoystickConnectEvent (sf) 

Event::JoystickMoveEvent (sf)

K

Keyboard (sf)
Event::KeyEvent (sf)

L

Listener (sf)
Ftp::ListingResponse (sf)
Lock (sf) 

M

MemoryInputStream (sf)

[A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [H](#) | [I](#) | [J](#) | [K](#) | [L](#) | [M](#) | [N](#) | [O](#) | [P](#) | [F](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

▼  sf::AIResource	Base class for classes that
 sf::SoundBuffer	Storage for audio samples c
►  sf::SoundRecorder	Abstract base class for capt
►  sf::SoundSource	Base class defining a sound
 sf::BlendMode	Blending modes for drawing
 sf::SoundStream::Chunk	Structure defining a chunk c
 sf::Clock	Utility class that measures t
 sf::Color	Utility class for manipulating
 sf::ContextSettings	Structure defining the settin
 sf::Shader::CurrentTextureType	Special type that can be pa
▼  sf::Drawable	represents the texture of the
►  sf::Shape	Abstract base class for obje
	target
	Base class for textured sha

sf::Sprite	Drawable representation of transformations, color, etc
sf::Text	Graphical text that can be drawn
sf::VertexArray	Define a set of one or more vertex arrays
sf::Event	Defines a system event and its properties
sf::Font	Class for loading and manipulating fonts
▼ sf::GIResource	Base class for classes that represent graphics resources
sf::Context	Class holding a valid drawing context
sf::Shader	Shader class (vertex, geometry, fragment)
sf::Texture	Image living on the graphics card
► sf::Window	Window that serves as a target for drawing
sf::Glyph	Structure describing a glyph
sf::Joystick::Identification	Structure holding a joystick's identification
sf::Image	Class for loading, manipulating and saving images
sf::SoundFileReader::Info	Structure holding the audio file information
sf::Font::Info	Holds various information about a font
▼ sf::InputStream	Abstract class for custom file input streams
sf::FileInputStream	Implementation of input stream reading from files
sf::MemoryInputStream	Implementation of input stream reading from memory
sf::IpAddress	Encapsulate an IPv4 network address
sf::Joystick	Give access to the real-time data of a joystick
sf::Event::JoystickButtonEvent	Joystick buttons events parameters (JoystickButtonPressed, JoystickButtonReleased)
sf::Event::JoystickConnectEvent	Joystick connection events (JoystickConnected, JoystickDisconnected)
sf::Event::JoystickMoveEvent	Joystick axis move event parameters (JoystickAxisMoved)

● sf::Keyboard	Give access to the real-time Keyboard event parameters
● sf::Event::KeyEvent	Keyboard event parameters
● sf::Listener	The audio listener is the point where the sounds are heard
● sf::Mouse	Give access to the real-time Mouse buttons events parameter (MouseButtonReleased)
● sf::Event::MouseEvent	Mouse move event parameters
● sf::Event::MouseWheelEvent	Mouse wheel events parameters
● sf::Event::MouseWheelScrollEvent	Mouse wheel events parameters
▼ ● sf::NonCopyable	Utility class that makes any class non-copyable
● sf::Context	Class holding a valid drawer
● sf::FileInputStream	Implementation of input stream
● sf::Ftp	A FTP client
● sf::GIResource::TransientContextLock	RAII helper class to temporally lock a resource
● sf::Http	A HTTP client
● sf::InputSoundFile	Provide read access to sound files
● sf::Lock	Automatic wrapper for locking
● sf::Mutex	Blocks concurrent access to shared resources by threads
● sf::OutputSoundFile	Provide write access to sound files
► ● sf::RenderTarget	Base class for all render targets
● sf::Shader	Shader class (vertex, geometry, fragment)
► ● sf::Socket	Base class for all the socket classes
● sf::Thread	Utility class to manipulate threads
► ● sf::ThreadLocal	Defines variables with thread local storage

 sf::Window	Window that serves as a ta
 sf::Packet	Utility class to build blocks c
 sf::Rect< T >	Utility class for manipulating
 sf::Rect< float >	
 sf::Rect< int >	
 sf::RenderStates	Define the states used for d
 sf::Http::Request	Define a HTTP request
 sf::Ftp::Response	Define a FTP response
 sf::Ftp::DirectoryResponse	Specialization of FTP respo
 sf::Ftp::ListingResponse	Specialization of FTP respo
 sf::Http::Response	Define a HTTP response
 sf::Sensor	Give access to the real-time
 sf::Event::SensorEvent	Sensor event parameters (S
 sf::Event::SizeEvent	Size events parameters (Re
 sf::SocketSelector	Multiplexer that allows to re
 sf::SoundFileFactory	Manages and instantiates so
 sf::SoundFileReader	Abstract base class for sour
 sf::SoundFileWriter	Abstract base class for sour
 sf::String	Utility string class that aut
 sf::Event::TextEvent	between types and encoding
 sf::Time	Text event parameters (Text
 sf::Touch	Represents a time value
 sf::Event::TouchEvent	Give access to the real-time
 sf::Transform	Touch events parameters (T
	TouchEnded)
	Define a 3x3 transform matr

▼  sf::Transformable	Decomposed transform definition for a scale
 sf::Shape	Base class for textured shapes
 sf::Sprite	Drawable representation of transformations, color, etc
 sf::Text	Graphical text that can be drawn
 sf::Utf< N >	Utility class providing general
 sf::Utf< 16 >	Specialization of the Utf template
 sf::Utf< 32 >	Specialization of the Utf template
 sf::Utf< 8 >	Specialization of the Utf template
 sf::Vector2< T >	Utility template class for mathematical
 sf::Vector2< float >	
 sf::Vector2< unsigned int >	
 sf::Vector3< T >	Utility template class for mathematical
 sf::Vertex	Define a point with color and texture coordinates
 sf::VideoMode	VideoMode defines a video mode
 sf::View	2D camera that defines what is visible

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- a -

- a : sf::Color
- A : sf::Keyboard
- Accelerometer : sf::Sensor
- accept() : sf::TcpListener
- Accepted : sf::Http::Response
- Add : sf::BlendMode , sf::Keyboard
- add() : sf::SocketSelector
- advance : sf::Glyph
- alphaDstFactor : sf::BlendMode
- alphaEquation : sf::BlendMode
- alphaSrcFactor : sf::BlendMode
- AIResource() : sf::AIResource
- alt : sf::Event::KeyEvent
- antialiasingLevel : sf::ContextSettings

- Any : sf::IpAddress
- AnyPort : sf::Socket
- append() : sf::Packet , sf::VertexArray
- Ascii : sf::Ftp
- asMicroseconds() : sf::Time
- asMilliseconds() : sf::Time
- asSeconds() : sf::Time
- Attribute : sf::ContextSettings
- attributeFlags : sf::ContextSettings
- axis : sf::Event::JoystickMoveEvent
- Axis : sf::Joystick
- AxisCount : sf::Joystick

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

Here is a list of all documented class members with links to the class documentation.

- b -

- [b : sf::Color](#)
- [B : sf::Keyboard](#)
- [BackSlash : sf::Keyboard](#)
- [BackSpace : sf::Keyboard](#)
- [BadCommandSequence : sf::Ftp::Response](#)
- [BadGateway : sf::Http::Response](#)
- [BadRequest : sf::Http::Response](#)
- [begin\(\) : sf::String](#)
- [Binary : sf::Ftp](#)
- [bind\(\) : sf::Shader , sf::Texture , sf::UdpSocket](#)
- [bitsPerPixel : sf::VideoMode](#)
- [Black : sf::Color](#)
- [BlendMode\(\) : sf::BlendMode](#)
- [blendMode : sf::RenderStates](#)

- Blue : sf::Color
 - Bold : sf::Text
 - bounds : sf::Glyph
 - Broadcast : sf::IpAddress
 - button : sf::Event::JoystickButtonEvent , sf::Event::MouseButtonEvent
 - Button : sf::Mouse
 - ButtonCount : sf::Joystick , sf::Mouse
-

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- C -

- [C : sf::Keyboard](#)
- [capture\(\) : sf::RenderWindow](#)
- [changeDirectory\(\) : sf::Ftp](#)
- [channelCount : sf::SoundFileReader::Info](#)
- [CircleShape\(\) : sf::CircleShape](#)
- [clear\(\) : sf::Packet , sf::RenderTarget , sf::SocketSelector , sf::String , sf::Vector2f](#)
- [Clock\(\) : sf::Clock](#)
- [close\(\) : sf::Socket , sf::TcpListener , sf::Window](#)
- [Closed : sf::Event](#)
- [ClosingConnection : sf::Ftp::Response](#)
- [ClosingDataConnection : sf::Ftp::Response](#)
- [code : sf::Event::KeyEvent](#)
- [Color\(\) : sf::Color](#)
- [color : sf::Vertex](#)

- colorDstFactor : sf::BlendMode
- colorEquation : sf::BlendMode
- colorSrcFactor : sf::BlendMode
- combine() : sf::Transform
- Comma : sf::Keyboard
- CommandNotImplemented : sf::Ftp::Response
- CommandUnknown : sf::Ftp::Response
- connect() : sf::Ftp , sf::TcpSocket
- ConnectionClosed : sf::Ftp::Response
- ConnectionFailed : sf::Ftp::Response , sf::Http::Response
- ConstIterator : sf::String
- contains() : sf::Rect< T >
- Context() : sf::Context
- ContextSettings() : sf::ContextSettings
- control : sf::Event::KeyEvent
- ConvexShape() : sf::ConvexShape
- CoordinateType : sf::Texture
- copy() : sf::Image
- copyToImage() : sf::Texture
- Core : sf::ContextSettings
- Count : sf::Event , sf::Joystick , sf::Sensor
- count() : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >
- create() : sf::Image , sf::RenderTexture , sf::Socket , sf::Texture , sf::View
- Created : sf::Http::Response
- createDirectory() : sf::Ftp
- createMaskFromColor() : sf::Image

- `createReaderFromFilename()` : `sf::SoundFileFactory`
 - `createReaderFromMemory()` : `sf::SoundFileFactory`
 - `createReaderFromStream()` : `sf::SoundFileFactory`
 - `createWriterFromFilename()` : `sf::SoundFileFactory`
 - `CurrentTexture` : `sf::Shader`
 - `Cyan` : `sf::Color`
-

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- d -

- D : [sf::Keyboard](#)
- Dash : [sf::Keyboard](#)
- DataConnectionAlreadyOpened : [sf::Ftp::Response](#)
- DataConnectionOpened : [sf::Ftp::Response](#)
- DataConnectionUnavailable : [sf::Ftp::Response](#)
- Debug : [sf::ContextSettings](#)
- decode() : [sf::Utf< 16 >](#) , [sf::Utf< 32 >](#) , [sf::Utf< 8 >](#)
- decodeAnsi() : [sf::Utf< 32 >](#)
- decodeWide() : [sf::Utf< 32 >](#)
- Default : [sf::ContextSettings](#) , [sf::RenderStates](#)
- Delete : [sf::Http::Request](#) , [sf::Keyboard](#)
- deleteDirectory() : [sf::Ftp](#)
- deleteFile() : [sf::Ftp](#)
- delta : [sf::Event::MouseWheelEvent](#) , [sf::Event::MouseWheelScrollViewEvent](#)

- depthBits : sf::ContextSettings
- DirectoryOk : sf::Ftp::Response
- DirectoryResponse() : sf::Ftp::DirectoryResponse
- DirectoryStatus : sf::Ftp::Response
- disconnect() : sf::Ftp , sf::TcpSocket
- Disconnected : sf::Socket
- display() : sf::RenderTexture , sf::Window
- Divide : sf::Keyboard
- Done : sf::Socket
- Down : sf::Keyboard
- download() : sf::Ftp
- draw() : sf::Drawable , sf::RenderTarget
- DstAlpha : sf::BlendMode
- DstColor : sf::BlendMode

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- e -

- E : [sf::Keyboard](#)
- Ebcdic : [sf::Ftp](#)
- encode() : [sf::Utf< 16 >](#) , [sf::Utf< 32 >](#) , [sf::Utf< 8 >](#)
- encodeAnsi() : [sf::Utf< 32 >](#)
- encodeWide() : [sf::Utf< 32 >](#)
- End : [sf::Keyboard](#)
- end() : [sf::String](#)
- endOfPacket() : [sf::Packet](#)
- ensureGIContext() : [sf::GIResource](#)
- EnteringPassiveMode : [sf::Ftp::Response](#)
- Equal : [sf::Keyboard](#)
- Equation : [sf::BlendMode](#)
- erase() : [sf::String](#)
- Error : [sf::Socket](#)

- Escape : sf::Keyboard
 - EventType : sf::Event
-

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- f -

- [F : sf::Keyboard](#)
- [F1 : sf::Keyboard](#)
- [F10 : sf::Keyboard](#)
- [F11 : sf::Keyboard](#)
- [F12 : sf::Keyboard](#)
- [F13 : sf::Keyboard](#)
- [F14 : sf::Keyboard](#)
- [F15 : sf::Keyboard](#)
- [F2 : sf::Keyboard](#)
- [F3 : sf::Keyboard](#)
- [F4 : sf::Keyboard](#)
- [F5 : sf::Keyboard](#)
- [F6 : sf::Keyboard](#)
- [F7 : sf::Keyboard](#)

- F8 : sf::Keyboard
- F9 : sf::Keyboard
- Factor : sf::BlendMode
- family : sf::Font::Info
- FileActionAborted : sf::Ftp::Response
- FileActionOk : sf::Ftp::Response
- FileInputStream() : sf::FileInputStream
- FilenameNotAllowed : sf::Ftp::Response
- FileStatus : sf::Ftp::Response
- FileUnavailable : sf::Ftp::Response
- find() : sf::String
- findCharacterPos() : sf::Text
- finger : sf::Event::TouchEvent
- flipHorizontally() : sf::Image
- flipVertically() : sf::Image
- Font() : sf::Font
- Forbidden : sf::Http::Response
- Fragment : sf::Shader
- fromAnsi() : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >
- fromLatin1() : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >
- fromUtf16() : sf::String
- fromUtf32() : sf::String
- fromUtf8() : sf::String
- fromWide() : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- g -

- [g : sf::Color](#)
- [G : sf::Keyboard](#)
- [GainedFocus : sf::Event](#)
- [GatewayTimeout : sf::Http::Response](#)
- [generateMipmap\(\) : sf::RenderTexture , sf::Texture](#)
- [Geometry : sf::Shader](#)
- [Get : sf::Http::Request](#)
- [getActiveContext\(\) : sf::Context](#)
- [getAttenuation\(\) : sf::SoundSource](#)
- [getAvailableDevices\(\) : sf::SoundRecorder](#)
- [getAxisPosition\(\) : sf::Joystick](#)
- [getBody\(\) : sf::Http::Response](#)
- [getBounds\(\) : sf::VertexArray](#)
- [getBuffer\(\) : sf::Sound , sf::SoundBufferRecorder](#)

- `getButtonCount() : sf::Joystick`
- `getCenter() : sf::View`
- `getChannelCount() : sf::InputSoundFile , sf::SoundBuffer , sf::SoundR`
- `getCharacterSize() : sf::Text`
- `getColor() : sf::Sprite , sf::Text`
- `getData() : sf::Packet , sf::String`
- `getDataSize() : sf::Packet`
- `getDefaultDevice() : sf::SoundRecorder`
- `getDefaultView() : sf::RenderTarget`
- `getDesktopMode() : sf::VideoMode`
- `getDevice() : sf::SoundRecorder`
- `getDirection() : sf::Listener`
- `getDirectory() : sf::Ftp::DirectoryResponse`
- `getDirectoryListing() : sf::Ftp`
- `getDuration() : sf::InputSoundFile , sf::Music , sf::SoundBuffer`
- `getElapsedTime() : sf::Clock`
- `getField() : sf::Http::Response`
- `getFillColor() : sf::Shape , sf::Text`
- `getFont() : sf::Text`
- `getFullscreenModes() : sf::VideoMode`
- `getFunction() : sf::Context`
- `getGlobalBounds() : sf::Shape , sf::Sprite , sf::Text`
- `getGlobalVolume() : sf::Listener`
- `getGlyph() : sf::Font`
- `getHandle() : sf::Socket`
- `getIdentification() : sf::Joystick`

- `getInfo() : sf::Font`
- `getInverse() : sf::Transform`
- `getInverseTransform() : sf::Transformable , sf::View`
- `getKerning() : sf::Font`
- `getLineSpacing() : sf::Font`
- `getListing() : sf::Ftp::ListingResponse`
- `getLocalAddress() : sf::IpAddress`
- `getLocalBounds() : sf::Shape , sf::Sprite , sf::Text`
- `getLocalPort() : sf::TcpListener , sf::TcpSocket , sf::UdpSocket`
- `getLoop() : sf::Sound , sf::SoundStream`
- `getMajorHttpVersion() : sf::Http::Response`
- `getMatrix() : sf::Transform`
- `getMaximumSize() : sf::Texture`
- `getMessage() : sf::Ftp::Response`
- `getMinDistance() : sf::SoundSource`
- `getMinorHttpVersion() : sf::Http::Response`
- `getNativeHandle() : sf::Shader , sf::Texture`
- `getOrigin() : sf::Transformable`
- `getOutlineColor() : sf::Shape , sf::Text`
- `getOutlineThickness() : sf::Shape , sf::Text`
- `getPitch() : sf::SoundSource`
- `getPixel() : sf::Image`
- `getPixelsPtr() : sf::Image`
- `getPlayingOffset() : sf::Sound , sf::SoundStream`
- `getPoint() : sf::CircleShape , sf::ConvexShape , sf::RectangleShape ,`
- `getPointCount() : sf::CircleShape , sf::ConvexShape , sf::RectangleSh`

- `getPosition() : sf::Listener , sf::Mouse , sf::SoundSource , sf::Touch , sf::Transformable`
- `getPrimitiveType() : sf::VertexArray`
- `getPublicAddress() : sf::IpAddress`
- `getRadius() : sf::CircleShape`
- `getRemoteAddress() : sf::TcpSocket`
- `getRemotePort() : sf::TcpSocket`
- `getRotation() : sf::Transformable , sf::View`
- `getSampleCount() : sf::InputSoundFile , sf::SoundBuffer`
- `getSampleRate() : sf::InputSoundFile , sf::SoundBuffer , sf::SoundRecorder`
- `getSamples() : sf::SoundBuffer`
- `getScale() : sf::Transformable`
- `getSettings() : sf::Context , sf::Window`
- `getSize() : sf::FileInputStream , sf::Image , sf::InputStream , sf::Memo , sf::RectangleShape , sf::RenderTarget , sf::RenderTexture , sf::RenderTarget , sf::View , sf::Window`
- `getStatus() : sf::Ftp::Response , sf::Http::Response , sf::Sound , sf::Sound`
- `getString() : sf::Text`
- `getStyle() : sf::Text`
- `getSystemHandle() : sf::Window`
- `getTexture() : sf::Font , sf::RenderTexture , sf::Shape , sf::Sprite`
- `getTextureRect() : sf::Shape , sf::Sprite`
- `getTransform() : sf::Transformable , sf::View`
- `getUnderlinePosition() : sf::Font`
- `getUnderlineThickness() : sf::Font`
- `getUpVector() : sf::Listener`
- `getValue() : sf::Sensor , sf::ThreadLocal`
- `getVertexCount() : sf::VertexArray`

- `getView() : sf::RenderTarget`
 - `getViewport() : sf::RenderTarget , sf::View`
 - `getVolume() : sf::SoundSource`
 - `getWorkingDirectory() : sf::Ftp`
 - `GIResource() : sf::GIResource`
 - `Glyph() : sf::Glyph`
 - `Gravity : sf::Sensor`
 - `Green : sf::Color`
 - `Gyroscope : sf::Sensor`
-

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- h -

- H : [sf::Keyboard](#)
- hasAxis() : [sf::Joystick](#)
- hasFocus() : [sf::Window](#)
- Head : [sf::Http::Request](#)
- height : [sf::Event::SizeEvent](#) , [sf::Rect< T >](#) , [sf::VideoMode](#)
- HelpMessage : [sf::Ftp::Response](#)
- Home : [sf::Keyboard](#)
- HorizontalWheel : [sf::Mouse](#)
- Http() : [sf::Http](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- i -

- I : [sf::Keyboard](#)
- Identity : [sf::Transform](#)
- Image() : [sf::Image](#)
- initialize() : [sf::RenderTarget](#) , [sf::SoundStream](#)
- InputSoundFile() : [sf::InputSoundFile](#)
- Insert : [sf::Keyboard](#)
- insert() : [sf::String](#)
- InsufficientStorageSpace : [sf::Ftp::Response](#)
- InternalServerError : [sf::Http::Response](#)
- intersects() : [sf::Rect< T >](#)
- InvalidFile : [sf::Ftp::Response](#)
- InvalidPos : [sf::String](#)
- InvalidResponse : [sf::Ftp::Response](#) , [sf::Http::Response](#)
- IpAddress() : [sf::IpAddress](#)

- `isAvailable() : sf::Sensor , sf::Shader , sf::SoundRecorder`
- `isBlocking() : sf::Socket`
- `isButtonPressed() : sf::Joystick , sf::Mouse`
- `isConnected() : sf::Joystick`
- `isDown() : sf::Touch`
- `isEmpty() : sf::String`
- `isExtensionAvailable() : sf::Context`
- `isGeometryAvailable() : sf::Shader`
- `isKeyPressed() : sf::Keyboard`
- `isOk() : sf::Ftp::Response`
- `isOpen() : sf::Window`
- `isReady() : sf::SocketSelector`
- `isRelativeToListener() : sf::SoundSource`
- `isRepeated() : sf::RenderTexture , sf::Texture`
- `isSmooth() : sf::RenderTexture , sf::Texture`
- `isSrgb() : sf::Texture`
- `isValid() : sf::VideoMode`
- `Italic : sf::Text`
- `Iterator : sf::String`

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- j -

- J : [sf::Keyboard](#)
- joystickButton : [sf::Event](#)
- JoystickButtonPressed : [sf::Event](#)
- JoystickButtonReleased : [sf::Event](#)
- joystickConnect : [sf::Event](#)
- JoystickConnected : [sf::Event](#)
- JoystickDisconnected : [sf::Event](#)
- joystickId : [sf::Event::JoystickButtonEvent](#) , [sf::Event::JoystickConnectEvent](#) , [sf::Event::JoystickMoveEvent](#)
- joystickMove : [sf::Event](#)
- JoystickMoved : [sf::Event](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- k -

- K : [sf::Keyboard](#)
- keepAlive() : [sf::Ftp](#)
- key : [sf::Event](#)
- Key : [sf::Keyboard](#)
- KeyCount : [sf::Keyboard](#)
- KeyPressed : [sf::Event](#)
- KeyReleased : [sf::Event](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- | -

- L : [sf::Keyboard](#)
- LAlt : [sf::Keyboard](#)
- launch() : [sf::Thread](#)
- LBracket : [sf::Keyboard](#)
- LControl : [sf::Keyboard](#)
- Left : [sf::Keyboard](#) , [sf::Mouse](#)
- left : [sf::Rect< T >](#)
- listen() : [sf::TcpListener](#)
- ListingResponse() : [sf::Ftp::ListingResponse](#)
- loadFromFile() : [sf::Font](#) , [sf::Image](#) , [sf::Shader](#) , [sf::SoundBuffer](#) , [sf::Texture](#)
- loadFromImage() : [sf::Texture](#)
- loadFromMemory() : [sf::Font](#) , [sf::Image](#) , [sf::Shader](#) , [sf::SoundBuffer](#)
- loadFromSamples() : [sf::SoundBuffer](#)
- loadFromStream() : [sf::Font](#) , [sf::Image](#) , [sf::Shader](#) , [sf::SoundBuffer](#)

- LocalError : sf::Ftp::Response
 - LocalHost : sf::IpAddress
 - Lock() : sf::Lock
 - lock() : sf::Mutex
 - LoggedIn : sf::Ftp::Response
 - login() : sf::Ftp
 - LostFocus : sf::Event
 - LShift : sf::Keyboard
 - LSystem : sf::Keyboard
-

SFML is licensed under the terms and conditions of the zlib/png
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- m -

- M : [sf::Keyboard](#)
- m_source : [sf::SoundSource](#)
- Magenta : [sf::Color](#)
- Magnetometer : [sf::Sensor](#)
- majorVersion : [sf::ContextSettings](#)
- mapCoordsToPixel() : [sf::RenderTarget](#)
- mapPixelToCoords() : [sf::RenderTarget](#)
- MaxDatagramSize : [sf::UdpSocket](#)
- MemoryInputStream() : [sf::MemoryInputStream](#)
- Menu : [sf::Keyboard](#)
- Method : [sf::Http::Request](#)
- microseconds() : [sf::Time](#)
- Middle : [sf::Mouse](#)
- milliseconds() : [sf::Time](#)

- minorVersion : sf::ContextSettings
- mouseButton : sf::Event
- MouseButtonPressed : sf::Event
- MouseButtonReleased : sf::Event
- MouseEntered : sf::Event
- MouseLeft : sf::Event
- mouseMove : sf::Event
- MouseMoved : sf::Event
- mouseWheel : sf::Event
- MouseWheelMoved : sf::Event
- mouseWheelScroll : sf::Event
- MouseWheelScrolled : sf::Event
- move() : sf::Transformable , sf::View
- MovedPermanently : sf::Http::Response
- MovedTemporarily : sf::Http::Response
- MultipleChoices : sf::Http::Response
- Multiply : sf::Keyboard
- Music() : sf::Music
- Mutex() : sf::Mutex

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- n -

- N : [sf::Keyboard](#)
- name : [sf::Joystick::Identification](#)
- NeedAccountToLogin : [sf::Ftp::Response](#)
- NeedAccountToStore : [sf::Ftp::Response](#)
- NeedInformation : [sf::Ftp::Response](#)
- NeedPassword : [sf::Ftp::Response](#)
- next() : [sf::Utf< 16 >](#) , [sf::Utf< 32 >](#) , [sf::Utf< 8 >](#)
- NoContent : [sf::Http::Response](#)
- NonCopyable() : [sf::NonCopyable](#)
- None : [sf::IpAddress](#)
- Normalized : [sf::Texture](#)
- NotEnoughMemory : [sf::Ftp::Response](#)
- NotFound : [sf::Http::Response](#)
- NotImplemented : [sf::Http::Response](#)

- NotLoggedIn : sf::Ftp::Response
- NotModified : sf::Http::Response
- NotReady : sf::Socket
- Num0 : sf::Keyboard
- Num1 : sf::Keyboard
- Num2 : sf::Keyboard
- Num3 : sf::Keyboard
- Num4 : sf::Keyboard
- Num5 : sf::Keyboard
- Num6 : sf::Keyboard
- Num7 : sf::Keyboard
- Num8 : sf::Keyboard
- Num9 : sf::Keyboard
- Numpad0 : sf::Keyboard
- Numpad1 : sf::Keyboard
- Numpad2 : sf::Keyboard
- Numpad3 : sf::Keyboard
- Numpad4 : sf::Keyboard
- Numpad5 : sf::Keyboard
- Numpad6 : sf::Keyboard
- Numpad7 : sf::Keyboard
- Numpad8 : sf::Keyboard
- Numpad9 : sf::Keyboard

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- O -

- O : [sf::Keyboard](#)
- Ok : [sf::Ftp::Response](#) , [sf::Http::Response](#)
- onCreate() : [sf::RenderWindow](#) , [sf::Window](#)
- One : [sf::BlendMode](#)
- OneMinusDstAlpha : [sf::BlendMode](#)
- OneMinusDstColor : [sf::BlendMode](#)
- OneMinusSrcAlpha : [sf::BlendMode](#)
- OneMinusSrcColor : [sf::BlendMode](#)
- onGetData() : [sf::Music](#) , [sf::SoundStream](#)
- onProcessSamples() : [sf::SoundBufferRecorder](#) , [sf::SoundRecorder](#)
- onReceive() : [sf::Packet](#)
- onResize() : [sf::RenderWindow](#) , [sf::Window](#)
- onSeek() : [sf::Music](#) , [sf::SoundStream](#)
- onSend() : [sf::Packet](#)

- onStart() : sf::SoundBufferRecorder , sf::SoundRecorder
- onStop() : sf::SoundBufferRecorder , sf::SoundRecorder
- open() : sf::FileInputStream , sf::MemoryInputStream , sf::SoundFileR
- openForWriting() : sf::InputSoundFile
- openFromFile() : sf::InputSoundFile , sf::Music , sf::OutputSoundFile
- openFromMemory() : sf::InputSoundFile , sf::Music
- openFromStream() : sf::InputSoundFile , sf::Music
- OpeningDataConnection : sf::Ftp::Response
- operator BoolType() : sf::Packet
- operator std::string() : sf::String
- operator std::wstring() : sf::String
- operator T *() : sf::ThreadLocalPtr< T >
- operator!=() : sf::BlendMode , sf::Color , sf::Rect< T > , sf::String , sf::Vector3< T > , sf::VideoMode
- operator%() : sf::Time
- operator%=() : sf::Time
- operator*() : sf::Color , sf::ThreadLocalPtr< T > , sf::Time , sf::Transfo
- operator*=() : sf::Color , sf::ThreadLocalPtr< T > , sf::Time , sf::Transfo
- operator+() : sf::Color , sf::String , sf::Time , sf::Vector2< T > , sf::Vect
- operator+=() : sf::Color , sf::String , sf::Time , sf::Vector2< T > , sf::Vect
- operator-() : sf::Color , sf::Time , sf::Vector2< T > , sf::Vector3< T >
- operator-=() : sf::Color , sf::Time , sf::Vector2< T > , sf::Vector3< T >
- operator->() : sf::ThreadLocalPtr< T >
- operator/() : sf::Time , sf::Vector2< T > , sf::Vector3< T >
- operator/=() : sf::Time , sf::Vector2< T > , sf::Vector3< T >
- operator< : sf::IpAddress , sf::String , sf::Time , sf::VideoMode

- `operator<<() : sf::Packet`
- `operator<=() : sf::String , sf::Time , sf::VideoMode`
- `operator=() : sf::Font , sf::SocketSelector , sf::Sound , sf::SoundBuffer
sf::Texture , sf::ThreadLocalPtr< T >`
- `operator==() : sf::BlendMode , sf::Color , sf::Rect< T > , sf::String , sf:
sf::Vector3< T > , sf::VideoMode`
- `operator>() : sf::String , sf::Time , sf::VideoMode`
- `operator>=() : sf::String , sf::Time , sf::VideoMode`
- `operator>>() : sf::Packet`
- `operator[]() : sf::String , sf::VertexArray`
- `Orientation : sf::Sensor`
- `OutputSoundFile() : sf::OutputSoundFile`

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- p -

- P : [sf::Keyboard](#)
- Packet() : [sf::Packet](#)
- PageDown : [sf::Keyboard](#)
- PageTypeUnknown : [sf::Ftp::Response](#)
- PageUp : [sf::Keyboard](#)
- ParameterNotImplemented : [sf::Ftp::Response](#)
- ParametersUnknown : [sf::Ftp::Response](#)
- parentDirectory() : [sf::Ftp](#)
- Partial : [sf::Socket](#)
- PartialContent : [sf::Http::Response](#)
- Pause : [sf::Keyboard](#)
- pause() : [sf::Sound](#) , [sf::SoundStream](#)
- Paused : [sf::SoundSource](#)
- Period : [sf::Keyboard](#)

- Pixels : sf::Texture
- play() : sf::Sound , sf::SoundStream
- Playing : sf::SoundSource
- PointlessCommand : sf::Ftp::Response
- pollEvent() : sf::Window
- popGLStates() : sf::RenderTarget
- position : sf::Event::JoystickMoveEvent , sf::Vertex
- Post : sf::Http::Request
- PovX : sf::Joystick
- PovY : sf::Joystick
- productId : sf::Joystick::Identification
- pushGLStates() : sf::RenderTarget
- Put : sf::Http::Request

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- q -

- Q : [sf::Keyboard](#)
- Quote : [sf::Keyboard](#)

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- r -

- [r : sf::Color](#)
- [R : sf::Joystick , sf::Keyboard](#)
- [RAlt : sf::Keyboard](#)
- [RangeNotSatisfiable : sf::Http::Response](#)
- [RBracket : sf::Keyboard](#)
- [RControl : sf::Keyboard](#)
- [read\(\) : sf::FileInputStream , sf::InputSoundFile , sf::InputStream , sf::SoundFileReader](#)
- [receive\(\) : sf::TcpSocket , sf::UdpSocket](#)
- [Rect\(\) : sf::Rect< T >](#)
- [RectangleShape\(\) : sf::RectangleShape](#)
- [Red : sf::Color](#)
- [registerReader\(\) : sf::SoundFileFactory](#)
- [registerWriter\(\) : sf::SoundFileFactory](#)

- Regular : sf::Text
 - remove() : sf::SocketSelector
 - renameFile() : sf::Ftp
 - RenderStates() : sf::RenderStates
 - RenderTarget() : sf::RenderTarget
 - RenderTexture() : sf::RenderTexture
 - RenderWindow() : sf::RenderWindow
 - replace() : sf::String
 - Request() : sf::Http::Request
 - requestFocus() : sf::Window
 - reset() : sf::View
 - resetBuffer() : sf::Sound
 - ResetContent : sf::Http::Response
 - resetGLStates() : sf::RenderTarget
 - resize() : sf::VertexArray
 - Resized : sf::Event
 - Response() : sf::Ftp::Response , sf::Http::Response
 - restart() : sf::Clock
 - RestartMarkerReply : sf::Ftp::Response
 - Return : sf::Keyboard
 - ReverseSubtract : sf::BlendMode
 - Right : sf::Keyboard , sf::Mouse
 - rotate() : sf::Transform , sf::Transformable , sf::View
 - RShift : sf::Keyboard
 - RSystem : sf::Keyboard
-

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- S -

- [S : sf::Keyboard](#)
- [sampleCount : sf::SoundFileReader::Info , sf::SoundStream::Chunk](#)
- [sampleRate : sf::SoundFileReader::Info](#)
- [samples : sf::SoundStream::Chunk](#)
- [saveToFile\(\) : sf::Image , sf::SoundBuffer](#)
- [scale\(\) : sf::Transform , sf::Transformable](#)
- [seconds\(\) : sf::Time](#)
- [seek\(\) : sf::FileInputStream , sf::InputSoundFile , sf::InputStream , sf::SoundFileReader](#)
- [SemiColon : sf::Keyboard](#)
- [send\(\) : sf::TcpSocket , sf::UdpSocket](#)
- [sendCommand\(\) : sf::Ftp](#)
- [sendRequest\(\) : sf::Http](#)
- [sensor : sf::Event](#)

- SensorChanged : sf::Event
- ServiceNotAvailable : sf::Http::Response
- ServiceReady : sf::Ftp::Response
- ServiceReadySoon : sf::Ftp::Response
- ServiceUnavailable : sf::Ftp::Response
- setActive() : sf::Context , sf::RenderTexture , sf::Window
- setAttenuation() : sf::SoundSource
- setBlocking() : sf::Socket
- setBody() : sf::Http::Request
- setBuffer() : sf::Sound
- setCenter() : sf::View
- setChannelCount() : sf::SoundRecorder
- setCharacterSize() : sf::Text
- setColor() : sf::Sprite , sf::Text
- setDevice() : sf::SoundRecorder
- setDirection() : sf::Listener
- setEnabled() : sf::Sensor
- setField() : sf::Http::Request
- setFillColor() : sf::Shape , sf::Text
- setFont() : sf::Text
- setFramerateLimit() : sf::Window
- setGlobalVolume() : sf::Listener
- setHost() : sf::Http
- setHttpVersion() : sf::Http::Request
- setIcon() : sf::Window
- setJoystickThreshold() : sf::Window

- `setKeyRepeatEnabled() : sf::Window`
- `setLoop() : sf::Sound , sf::SoundStream`
- `setMethod() : sf::Http::Request`
- `setMinDistance() : sf::SoundSource`
- `setMouseCursorGrabbed() : sf::Window`
- `setMouseCursorVisible() : sf::Window`
- `setOrigin() : sf::Transformable`
- `setOutlineColor() : sf::Shape , sf::Text`
- `setOutlineThickness() : sf::Shape , sf::Text`
- `setParameter() : sf::Shader`
- `setPitch() : sf::SoundSource`
- `setPixel() : sf::Image`
- `setPlayingOffset() : sf::Sound , sf::SoundStream`
- `setPoint() : sf::ConvexShape`
- `setPointCount() : sf::CircleShape , sf::ConvexShape`
- `setPosition() : sf::Listener , sf::Mouse , sf::SoundSource , sf::Transformable`
- `setPrimitiveType() : sf::VertexArray`
- `setProcessingInterval() : sf::SoundRecorder`
- `setRadius() : sf::CircleShape`
- `setRelativeToListener() : sf::SoundSource`
- `setRepeated() : sf::RenderTexture , sf::Texture`
- `setRotation() : sf::Transformable , sf::View`
- `setScale() : sf::Transformable`
- `setSize() : sf::RectangleShape , sf::View , sf::Window`
- `setSmooth() : sf::RenderTexture , sf::Texture`
- `setSrgb() : sf::Texture`

- `setString() : sf::Text`
- `setStyle() : sf::Text`
- `setTexture() : sf::Shape , sf::Sprite`
- `setTextureRect() : sf::Shape , sf::Sprite`
- `setTitle() : sf::Window`
- `setUniform() : sf::Shader`
- `setUniformArray() : sf::Shader`
- `setUpVector() : sf::Listener`
- `setUri() : sf::Http::Request`
- `setValue() : sf::ThreadLocal`
- `setVerticalSyncEnabled() : sf::Window`
- `setView() : sf::RenderTarget`
- `setViewport() : sf::View`
- `setVirtualKeyboardVisible() : sf::Keyboard`
- `setVisible() : sf::Window`
- `setVolume() : sf::SoundSource`
- `shader : sf::RenderStates`
- `Shader() : sf::Shader`
- `Shape() : sf::Shape`
- `shift : sf::Event::KeyEvent`
- `size : sf::Event`
- `Slash : sf::Keyboard`
- `Socket() : sf::Socket`
- `SocketSelector() : sf::SocketSelector`
- `Sound() : sf::Sound`
- `SoundBuffer() : sf::SoundBuffer`

- SoundRecorder() : sf::SoundRecorder
- SoundSource() : sf::SoundSource
- SoundStream() : sf::SoundStream
- Space : sf::Keyboard
- Sprite() : sf::Sprite
- SrcAlpha : sf::BlendMode
- SrcColor : sf::BlendMode
- sRgbCapable : sf::ContextSettings
- start() : sf::SoundRecorder
- Status : sf::Ftp::Response , sf::Http::Response , sf::Socket , sf::Sound
- stencilBits : sf::ContextSettings
- stop() : sf::Sound , sf::SoundRecorder , sf::SoundStream
- Stopped : sf::SoundSource
- StrikeThrough : sf::Text
- String() : sf::String
- Style : sf::Text
- substring() : sf::String
- Subtract : sf::BlendMode , sf::Keyboard
- system : sf::Event::KeyEvent
- SystemStatus : sf::Ftp::Response
- SystemType : sf::Ftp::Response

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[**t**](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- t -

- T : [sf::Keyboard](#)
- Tab : [sf::Keyboard](#)
- Tcp : [sf::Socket](#)
- TcpListener() : [sf::TcpListener](#)
- TcpSocket() : [sf::TcpSocket](#)
- tell() : [sf::FileInputStream](#) , [sf::InputStream](#) , [sf::MemoryInputStream](#)
- terminate() : [sf::Thread](#)
- texCoords : [sf::Vertex](#)
- text : [sf::Event](#)
- Text() : [sf::Text](#)
- TextEntered : [sf::Event](#)
- texture : [sf::RenderStates](#)
- Texture() : [sf::Texture](#)
- textureRect : [sf::Glyph](#)

- Thread() : sf::Thread
- ThreadLocal() : sf::ThreadLocal
- ThreadLocalPtr() : sf::ThreadLocalPtr< T >
- Tilde : sf::Keyboard
- Time() : sf::Time
- toAnsi() : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >
- toAnsiString() : sf::String
- toInteger() : sf::Color , sf::IpAddress
- toLatin1() : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >
- top : sf::Rect< T >
- toString() : sf::IpAddress
- touch : sf::Event
- TouchBegan : sf::Event
- TouchEnded : sf::Event
- TouchMoved : sf::Event
- toUtf16() : sf::String , sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >
- toUtf32() : sf::String , sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >
- toUtf8() : sf::String , sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >
- toWide() : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >
- toWideString() : sf::String
- TransferAborted : sf::Ftp::Response
- TransferMode : sf::Ftp
- transform : sf::RenderStates
- Transform() : sf::Transform
- Transformable() : sf::Transformable
- transformPoint() : sf::Transform

- transformRect() : sf::Transform
 - TransientContextLock() : sf::GIResource::TransientContextLock
 - translate() : sf::Transform
 - Transparent : sf::Color
 - type : sf::Event::SensorEvent , sf::Event
 - Type : sf::Sensor , sf::Shader , sf::Socket
-

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[**u**](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- U -

- U : [sf::Joystick](#) , [sf::Keyboard](#)
- Udp : [sf::Socket](#)
- UdpSocket() : [sf::UdpSocket](#)
- Unauthorized : [sf::Http::Response](#)
- unbind() : [sf::UdpSocket](#)
- Underlined : [sf::Text](#)
- unicode : [sf::Event::TextEvent](#)
- Unknown : [sf::Keyboard](#)
- unlock() : [sf::Mutex](#)
- unregisterReader() : [sf::SoundFileFactory](#)
- unregisterWriter() : [sf::SoundFileFactory](#)
- Up : [sf::Keyboard](#)
- update() : [sf::Joystick](#) , [sf::Shape](#) , [sf::Texture](#)
- upload() : [sf::Ftp](#)

- UserAcceleration : sf::Sensor

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- V -

- V : [sf::Joystick](#) , [sf::Keyboard](#)
- Vector2() : [sf::Vector2< T >](#)
- Vector3() : [sf::Vector3< T >](#)
- vendorId : [sf::Joystick::Identification](#)
- VersionNotSupported : [sf::Http::Response](#)
- Vertex : [sf::Shader](#) , [sf::Vertex](#)
- VertexArray() : [sf::VertexArray](#)
- VerticalWheel : [sf::Mouse](#)
- VideoMode() : [sf::VideoMode](#)
- View() : [sf::View](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- W -

- W : [sf::Keyboard](#)
- wait() : [sf::SocketSelector](#) , [sf::Thread](#)
- waitEvent() : [sf::Window](#)
- wheel : [sf::Event::MouseWheelScrollEvent](#)
- Wheel : [sf::Mouse](#)
- White : [sf::Color](#)
- width : [sf::Event::SizeEvent](#) , [sf::Rect< T >](#) , [sf::VideoMode](#)
- Window() : [sf::Window](#)
- write() : [sf::OutputSoundFile](#) , [sf::SoundFileWriter](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- X -

- x : sf::Event::MouseEvent , sf::Event::MouseMoveEvent , sf::Event::MouseWheelScrollEvent , sf::Event::SensorEvent , sf::Event
- X : sf::Joystick , sf::Keyboard
- x : sf::Vector2< T > , sf::Vector3< T >
- XButton1 : sf::Mouse
- XButton2 : sf::Mouse

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- y -

- y : sf::Event::MouseEvent , sf::Event::MouseMoveEvent , sf::Event::MouseWheelScrollEvent , sf::Event::SensorEvent , sf::Event
- Y : sf::Joystick , sf::Keyboard
- y : sf::Vector2< T > , sf::Vector3< T >
- Yellow : sf::Color

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- Z -

- z : sf::Event::SensorEvent
- Z : sf::Joystick , sf::Keyboard
- z : sf::Vector3< T >
- Zero : sf::BlendMode , sf::Time
- zoom() : sf::View

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)[~](#)

Here is a list of all documented class members with links to the class documentation.

- ~ -

- [~AIResource\(\) : sf::AIResource](#)
- [~Context\(\) : sf::Context](#)
- [~Drawable\(\) : sf::Drawable](#)
- [~FileInputStream\(\) : sf::FileInputStream](#)
- [~Font\(\) : sf::Font](#)
- [~Ftp\(\) : sf::Ftp](#)
- [~GIResource\(\) : sf::GIResource](#)
- [~Image\(\) : sf::Image](#)
- [~InputSoundFile\(\) : sf::InputSoundFile](#)
- [~InputStream\(\) : sf::InputStream](#)
- [~Lock\(\) : sf::Lock](#)
- [~Music\(\) : sf::Music](#)
- [~Mutex\(\) : sf::Mutex](#)
- [~OutputSoundFile\(\) : sf::OutputSoundFile](#)

- ~Packet() : sf::Packet
- ~RenderTarget() : sf::RenderTarget
- ~RenderTexture() : sf::RenderTexture
- ~RenderWindow() : sf::RenderWindow
- ~Shader() : sf::Shader
- ~Shape() : sf::Shape
- ~Socket() : sf::Socket
- ~SocketSelector() : sf::SocketSelector
- ~Sound() : sf::Sound
- ~SoundBuffer() : sf::SoundBuffer
- ~SoundBufferRecorder() : sf::SoundBufferRecorder
- ~SoundFileReader() : sf::SoundFileReader
- ~SoundFileWriter() : sf::SoundFileWriter
- ~SoundRecorder() : sf::SoundRecorder
- ~SoundSource() : sf::SoundSource
- ~SoundStream() : sf::SoundStream
- ~Texture() : sf::Texture
- ~Thread() : sf::Thread
- ~ThreadLocal() : sf::ThreadLocal
- ~Transformable() : sf::Transformable
- ~TransientContextLock() : sf::GIResource::TransientContextLock
- ~Window() : sf::Window

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- a -

- [accept\(\) : sf::TcpListener](#)
- [add\(\) : sf::SocketSelector](#)
- [AIResource\(\) : sf::AIResource](#)
- [append\(\) : sf::Packet , sf::VertexArray](#)
- [asMicroseconds\(\) : sf::Time](#)
- [asMilliseconds\(\) : sf::Time](#)
- [asSeconds\(\) : sf::Time](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[**b**](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- b -

- [begin\(\) : sf::String](#)
- [bind\(\) : sf::Shader , sf::Texture , sf::UdpSocket](#)
- [BlendMode\(\) : sf::BlendMode](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- C -

- [capture\(\) : sf::RenderWindow](#)
- [changeDirectory\(\) : sf::Ftp](#)
- [CircleShape\(\) : sf::CircleShape](#)
- [clear\(\) : sf::Packet , sf::RenderTarget , sf::SocketSelector , sf::String ,](#)
- [Clock\(\) : sf::Clock](#)
- [close\(\) : sf::Socket , sf::TcpListener , sf::Window](#)
- [Color\(\) : sf::Color](#)
- [combine\(\) : sf::Transform](#)
- [connect\(\) : sf::Ftp , sf::TcpSocket](#)
- [contains\(\) : sf::Rect< T >](#)
- [Context\(\) : sf::Context](#)
- [ContextSettings\(\) : sf::ContextSettings](#)
- [ConvexShape\(\) : sf::ConvexShape](#)
- [copy\(\) : sf::Image](#)

- `copyToImage() : sf::Texture`
 - `count() : sf::Utf< 16 >, sf::Utf< 32 >, sf::Utf< 8 >`
 - `create() : sf::Image, sf::RenderTexture, sf::Socket, sf::Texture, sf::View`
 - `createDirectory() : sf::Ftp`
 - `createMaskFromColor() : sf::Image`
 - `createReaderFromFilename() : sf::SoundFileFactory`
 - `createReaderFromMemory() : sf::SoundFileFactory`
 - `createReaderFromStream() : sf::SoundFileFactory`
 - `createWriterFromFilename() : sf::SoundFileFactory`
-

SFML is licensed under the terms and conditions of the zlib/png
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- d -

- [decode\(\) : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >](#)
- [decodeAnsi\(\) : sf::Utf< 32 >](#)
- [decodeWide\(\) : sf::Utf< 32 >](#)
- [deleteDirectory\(\) : sf::Ftp](#)
- [deleteFile\(\) : sf::Ftp](#)
- [DirectoryResponse\(\) : sf::Ftp::DirectoryResponse](#)
- [disconnect\(\) : sf::Ftp , sf::TcpSocket](#)
- [display\(\) : sf::RenderTexture , sf::Window](#)
- [download\(\) : sf::Ftp](#)
- [draw\(\) : sf::Drawable , sf::RenderTarget](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[**e**](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- e -

- [encode\(\) : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >](#)
- [encodeAnsi\(\) : sf::Utf< 32 >](#)
- [encodeWide\(\) : sf::Utf< 32 >](#)
- [end\(\) : sf::String](#)
- [endOfPacket\(\) : sf::Packet](#)
- [ensureGIContext\(\) : sf::GIResource](#)
- [erase\(\) : sf::String](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- f -

- [FileInputStream\(\) : sf::FileInputStream](#)
- [find\(\) : sf::String](#)
- [findCharacterPos\(\) : sf::Text](#)
- [flipHorizontally\(\) : sf::Image](#)
- [flipVertically\(\) : sf::Image](#)
- [Font\(\) : sf::Font](#)
- [fromAnsi\(\) : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >](#)
- [fromLatin1\(\) : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >](#)
- [fromUtf16\(\) : sf::String](#)
- [fromUtf32\(\) : sf::String](#)
- [fromUtf8\(\) : sf::String](#)
- [fromWide\(\) : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- g -

- [generateMipmap\(\) : sf::RenderTexture , sf::Texture](#)
- [getActiveContext\(\) : sf::Context](#)
- [getAttenuation\(\) : sf::SoundSource](#)
- [getAvailableDevices\(\) : sf::SoundRecorder](#)
- [getAxisPosition\(\) : sf::Joystick](#)
- [getBody\(\) : sf::Http::Response](#)
- [getBounds\(\) : sf::VertexArray](#)
- [getBuffer\(\) : sf::Sound , sf::SoundBufferRecorder](#)
- [getButtonCount\(\) : sf::Joystick](#)
- [getCenter\(\) : sf::View](#)
- [getChannelCount\(\) : sf::InputSoundFile , sf::SoundBuffer , sf::SoundR](#)
- [getCharacterSize\(\) : sf::Text](#)
- [getColor\(\) : sf::Sprite , sf::Text](#)
- [getData\(\) : sf::Packet , sf::String](#)

- `getDataSize() : sf::Packet`
- `getDefaultDevice() : sf::SoundRecorder`
- `getDefaultView() : sf::RenderTarget`
- `getDesktopMode() : sf::VideoMode`
- `getDevice() : sf::SoundRecorder`
- `getDirection() : sf::Listener`
- `getDirectory() : sf::Ftp::DirectoryResponse`
- `getDirectoryListing() : sf::Ftp`
- `getDuration() : sf::InputSoundFile , sf::Music , sf::SoundBuffer`
- `getElapsed Time() : sf::Clock`
- `getField() : sf::Http::Response`
- `getFillColor() : sf::Shape , sf::Text`
- `getFont() : sf::Text`
- `getFullscreenModes() : sf::VideoMode`
- `getFunction() : sf::Context`
- `getGlobalBounds() : sf::Shape , sf::Sprite , sf::Text`
- `getGlobalVolume() : sf::Listener`
- `getGlyph() : sf::Font`
- `getHandle() : sf::Socket`
- `getIdentification() : sf::Joystick`
- `getInfo() : sf::Font`
- `getInverse() : sf::Transform`
- `getInverseTransform() : sf::Transformable , sf::View`
- `getKerning() : sf::Font`
- `getLineSpacing() : sf::Font`
- `getListing() : sf::Ftp::ListingResponse`

- `getLocalAddress() : sf::IpAddress`
- `getLocalBounds() : sf::Shape , sf::Sprite , sf::Text`
- `getLocalPort() : sf::TcpListener , sf::TcpSocket , sf::UdpSocket`
- `getLoop() : sf::Sound , sf::SoundStream`
- `getMajorHttpVersion() : sf::Http::Response`
- `getMatrix() : sf::Transform`
- `getMaximumSize() : sf::Texture`
- `getMessage() : sf::Ftp::Response`
- `getMinDistance() : sf::SoundSource`
- `getMinorHttpVersion() : sf::Http::Response`
- `getNativeHandle() : sf::Shader , sf::Texture`
- `getOrigin() : sf::Transformable`
- `getOutlineColor() : sf::Shape , sf::Text`
- `getOutlineThickness() : sf::Shape , sf::Text`
- `getPitch() : sf::SoundSource`
- `getPixel() : sf::Image`
- `getPixelsPtr() : sf::Image`
- `getPlayingOffset() : sf::Sound , sf::SoundStream`
- `getPoint() : sf::CircleShape , sf::ConvexShape , sf::RectangleShape , sf::Text`
- `getPointCount() : sf::CircleShape , sf::ConvexShape , sf::RectangleShape , sf::Text`
- `getPosition() : sf::Listener , sf::Mouse , sf::SoundSource , sf::Touch , sf::Text`
- `getPrimitiveType() : sf::VertexArray`
- `getPublicAddress() : sf::IpAddress`
- `getRadius() : sf::CircleShape`
- `getRemoteAddress() : sf::TcpSocket`
- `getRemotePort() : sf::TcpSocket`

- `getRotation() : sf::Transformable , sf::View`
- `getSampleCount() : sf::InputSoundFile , sf::SoundBuffer`
- `getSampleRate() : sf::InputSoundFile , sf::SoundBuffer , sf::SoundRec`
- `getSamples() : sf::SoundBuffer`
- `getScale() : sf::Transformable`
- `getSettings() : sf::Context , sf::Window`
- `getSize() : sf::FileInputStream , sf::Image , sf::InputStream , sf::Memo
sf::RectangleShape , sf::RenderTarget , sf::RenderTexture , sf::Rende
, sf::View , sf::Window`
- `getStatus() : sf::Ftp::Response , sf::Http::Response , sf::Sound , sf::So`
- `getString() : sf::Text`
- `getStyle() : sf::Text`
- `getSystemHandle() : sf::Window`
- `getTexture() : sf::Font , sf::RenderTexture , sf::Shape , sf::Sprite`
- `getTextureRect() : sf::Shape , sf::Sprite`
- `getTransform() : sf::Transformable , sf::View`
- `getUnderlinePosition() : sf::Font`
- `getUnderlineThickness() : sf::Font`
- `getUpVector() : sf::Listener`
- `getValue() : sf::Sensor , sf::ThreadLocal`
- `getVertexCount() : sf::VertexArray`
- `getView() : sf::RenderTarget`
- `getViewport() : sf::RenderTarget , sf::View`
- `getVolume() : sf::SoundSource`
- `getWorkingDirectory() : sf::Ftp`
- `GIResource() : sf::GIResource`
- `Glyph() : sf::Glyph`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[**h**](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- h -

- [hasAxis\(\) : sf::Joystick](#)
- [hasFocus\(\) : sf::Window](#)
- [Http\(\) : sf::Http](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- | -

- [Image\(\) : sf::Image](#)
- [initialize\(\) : sf::RenderTarget , sf::SoundStream](#)
- [InputSoundFile\(\) : sf::InputSoundFile](#)
- [insert\(\) : sf::String](#)
- [intersects\(\) : sf::Rect< T >](#)
- [IpAddress\(\) : sf::IpAddress](#)
- [isAvailable\(\) : sf::Sensor , sf::Shader , sf::SoundRecorder](#)
- [isBlocking\(\) : sf::Socket](#)
- [isButtonPressed\(\) : sf::Joystick , sf::Mouse](#)
- [isConnected\(\) : sf::Joystick](#)
- [isDown\(\) : sf::Touch](#)
- [isEmpty\(\) : sf::String](#)
- [isExtensionAvailable\(\) : sf::Context](#)
- [isGeometryAvailable\(\) : sf::Shader](#)

- `isKeyPressed() : sf::Keyboard`
 - `isOk() : sf::Ftp::Response`
 - `isOpen() : sf::Window`
 - `isReady() : sf::SocketSelector`
 - `isRelativeToListener() : sf::SoundSource`
 - `isRepeated() : sf::RenderTexture , sf::Texture`
 - `isSmooth() : sf::RenderTexture , sf::Texture`
 - `isSrgb() : sf::Texture`
 - `isValid() : sf::VideoMode`
-

SFML is licensed under the terms and conditions of the zlib/png
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[**k**](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- k -

- [keepAlive\(\) : sf::Ftp](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- | -

- [launch\(\) : sf::Thread](#)
- [listen\(\) : sf::TcpListener](#)
- [ListingResponse\(\) : sf::Ftp::ListingResponse](#)
- [loadFromFile\(\) : sf::Font , sf::Image , sf::Shader , sf::SoundBuffer , sf:](#)
- [loadFromImage\(\) : sf::Texture](#)
- [loadFromMemory\(\) : sf::Font , sf::Image , sf::Shader , sf::SoundBuffer](#)
- [loadFromSamples\(\) : sf::SoundBuffer](#)
- [loadFromStream\(\) : sf::Font , sf::Image , sf::Shader , sf::SoundBuffer](#)
- [Lock\(\) : sf::Lock](#)
- [lock\(\) : sf::Mutex](#)
- [login\(\) : sf::Ftp](#)

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- m -

- [mapCoordsToPixel\(\) : sf::RenderTarget](#)
- [mapPixelToCoords\(\) : sf::RenderTarget](#)
- [MemoryInputStream\(\) : sf::MemoryInputStream](#)
- [microseconds\(\) : sf::Time](#)
- [milliseconds\(\) : sf::Time](#)
- [move\(\) : sf::Transformable , sf::View](#)
- [Music\(\) : sf::Music](#)
- [Mutex\(\) : sf::Mutex](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- n -

- [next\(\) : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >](#)
- [NonCopyable\(\) : sf::NonCopyable](#)

SFML is licensed under the terms and conditions of the zlib/png

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- O -

- [onCreate\(\) : sf::RenderWindow , sf::Window](#)
- [onGetData\(\) : sf::Music , sf::SoundStream](#)
- [onProcessSamples\(\) : sf::SoundBufferRecorder , sf::SoundRecorder](#)
- [onReceive\(\) : sf::Packet](#)
- [onResize\(\) : sf::RenderWindow , sf::Window](#)
- [onSeek\(\) : sf::Music , sf::SoundStream](#)
- [onSend\(\) : sf::Packet](#)
- [onStart\(\) : sf::SoundBufferRecorder , sf::SoundRecorder](#)
- [onStop\(\) : sf::SoundBufferRecorder , sf::SoundRecorder](#)
- [open\(\) : sf::FileInputStream , sf::MemoryInputStream , sf::SoundFileR](#)
- [openForWriting\(\) : sf::InputSoundFile](#)
- [openFromFile\(\) : sf::InputSoundFile , sf::Music , sf::OutputSoundFile](#)
- [openFromMemory\(\) : sf::InputSoundFile , sf::Music](#)
- [openFromStream\(\) : sf::InputSoundFile , sf::Music](#)

- operator BoolType() : sf::Packet
- operator std::string() : sf::String
- operator std::wstring() : sf::String
- operator T *() : sf::ThreadLocalPtr< T >
- operator!=() : sf::BlendMode , sf::Color , sf::Rect< T > , sf::String , sf::Vector3< T > , sf::VideoMode
- operator%() : sf::Time
- operator%=() : sf::Time
- operator*() : sf::Color , sf::ThreadLocalPtr< T > , sf::Time , sf::Transform , sf::Vector3< T >
- operator*=() : sf::Color , sf::Time , sf::Transform , sf::Vector2< T > , sf::Vector3< T >
- operator+() : sf::Color , sf::String , sf::Time , sf::Vector2< T > , sf::Vector3< T >
- operator+=() : sf::Color , sf::String , sf::Time , sf::Vector2< T > , sf::Vector3< T >
- operator-() : sf::Color , sf::Time , sf::Vector2< T > , sf::Vector3< T >
- operator-=() : sf::Color , sf::Time , sf::Vector2< T > , sf::Vector3< T >
- operator->() : sf::ThreadLocalPtr< T >
- operator/() : sf::Time , sf::Vector2< T > , sf::Vector3< T >
- operator/=() : sf::Time , sf::Vector2< T > , sf::Vector3< T >
- operator<() : sf::String , sf::Time , sf::VideoMode
- operator<<() : sf::Packet
- operator<=() : sf::String , sf::Time , sf::VideoMode
- operator=() : sf::Font , sf::SocketSelector , sf::Sound , sf::SoundBuffer , sf::Texture , sf::ThreadLocalPtr< T >
- operator==() : sf::BlendMode , sf::Color , sf::Rect< T > , sf::String , sf::Vector3< T > , sf::VideoMode
- operator>() : sf::String , sf::Time , sf::VideoMode
- operator>=() : sf::String , sf::Time , sf::VideoMode

- `operator>>() : sf::Packet`
 - `operator[]() : sf::String , sf::VertexArray`
 - `OutputSoundFile() : sf::OutputSoundFile`
-

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- p -

- [Packet\(\) : sf::Packet](#)
- [parentDirectory\(\) : sf::Ftp](#)
- [pause\(\) : sf::Sound , sf::SoundStream](#)
- [play\(\) : sf::Sound , sf::SoundStream](#)
- [pollEvent\(\) : sf::Window](#)
- [popGLStates\(\) : sf::RenderTarget](#)
- [pushGLStates\(\) : sf::RenderTarget](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- r -

- [read\(\) : sf::FileInputStream , sf::InputSoundFile , sf::InputStream , sf::SoundFileReader](#)
- [receive\(\) : sf::TcpSocket , sf::UdpSocket](#)
- [Rect\(\) : sf::Rect< T >](#)
- [RectangleShape\(\) : sf::RectangleShape](#)
- [registerReader\(\) : sf::SoundFileFactory](#)
- [registerWriter\(\) : sf::SoundFileFactory](#)
- [remove\(\) : sf::SocketSelector](#)
- [renameFile\(\) : sf::Ftp](#)
- [RenderStates\(\) : sf::RenderStates](#)
- [RenderTarget\(\) : sf::RenderTarget](#)
- [RenderTexture\(\) : sf::RenderTexture](#)
- [RenderWindow\(\) : sf::RenderWindow](#)
- [replace\(\) : sf::String](#)

- Request() : sf::Http::Request
 - requestFocus() : sf::Window
 - reset() : sf::View
 - resetBuffer() : sf::Sound
 - resetGLStates() : sf::RenderTarget
 - resize() : sf::VertexArray
 - Response() : sf::Ftp::Response , sf::Http::Response
 - restart() : sf::Clock
 - rotate() : sf::Transform , sf::Transformable , sf::View
-

SFML is licensed under the terms and conditions of the zlib/png
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- S -

- [saveToFile\(\) : sf::Image , sf::SoundBuffer](#)
- [scale\(\) : sf::Transform , sf::Transformable](#)
- [seconds\(\) : sf::Time](#)
- [seek\(\) : sf::FileInputStream , sf::InputSoundFile , sf::InputStream , sf::SoundFileReader](#)
- [send\(\) : sf::TcpSocket , sf::UdpSocket](#)
- [sendCommand\(\) : sf::Ftp](#)
- [sendRequest\(\) : sf::Http](#)
- [setActive\(\) : sf::Context , sf::RenderTexture , sf::Window](#)
- [setAttenuation\(\) : sf::SoundSource](#)
- [setBlocking\(\) : sf::Socket](#)
- [setBody\(\) : sf::Http::Request](#)
- [setBuffer\(\) : sf::Sound](#)
- [setCenter\(\) : sf::View](#)

- `setChannelCount() : sf::SoundRecorder`
- `setCharacterSize() : sf::Text`
- `setColor() : sf::Sprite , sf::Text`
- `setDevice() : sf::SoundRecorder`
- `setDirection() : sf::Listener`
- `setEnabled() : sf::Sensor`
- `setField() : sf::Http::Request`
- `setFillColor() : sf::Shape , sf::Text`
- `setFont() : sf::Text`
- `setFramerateLimit() : sf::Window`
- `setGlobalVolume() : sf::Listener`
- `setHost() : sf::Http`
- `setHttpVersion() : sf::Http::Request`
- `setIcon() : sf::Window`
- `setJoystickThreshold() : sf::Window`
- `setKeyRepeatEnabled() : sf::Window`
- `setLoop() : sf::Sound , sf::SoundStream`
- `setMethod() : sf::Http::Request`
- `setMinDistance() : sf::SoundSource`
- `setMouseCursorGrabbed() : sf::Window`
- `setMouseCursorVisible() : sf::Window`
- `setOrigin() : sf::Transformable`
- `setOutlineColor() : sf::Shape , sf::Text`
- `setOutlineThickness() : sf::Shape , sf::Text`
- `setParameter() : sf::Shader`
- `setPitch() : sf::SoundSource`

- `setPixel() : sf::Image`
- `setPlayingOffset() : sf::Sound , sf::SoundStream`
- `setPoint() : sf::ConvexShape`
- `setPointCount() : sf::CircleShape , sf::ConvexShape`
- `setPosition() : sf::Listener , sf::Mouse , sf::SoundSource , sf::Transform`
- `setPrimitiveType() : sf::VertexArray`
- `setProcessingInterval() : sf::SoundRecorder`
- `setRadius() : sf::CircleShape`
- `setRelativeToListener() : sf::SoundSource`
- `setRepeated() : sf::RenderTarget , sf::Texture`
- `setRotation() : sf::Transformable , sf::View`
- `setScale() : sf::Transformable`
- `setSize() : sf::RectangleShape , sf::View , sf::Window`
- `setSmooth() : sf::RenderTarget , sf::Texture`
- `setSrgb() : sf::Texture`
- `setString() : sf::Text`
- `setStyle() : sf::Text`
- `setTexture() : sf::Shape , sf::Sprite`
- `setTextureRect() : sf::Shape , sf::Sprite`
- `setTitle() : sf::Window`
- `setUniform() : sf::Shader`
- `setUniformArray() : sf::Shader`
- `setUpVector() : sf::Listener`
- `setUri() : sf::Http::Request`
- `setValue() : sf::ThreadLocal`
- `setVerticalSyncEnabled() : sf::Window`

- `setView() : sf::RenderTarget`
- `setViewport() : sf::View`
- `setVirtualKeyboardVisible() : sf::Keyboard`
- `setVisible() : sf::Window`
- `setVolume() : sf::SoundSource`
- `Shader() : sf::Shader`
- `Shape() : sf::Shape`
- `Socket() : sf::Socket`
- `SocketSelector() : sf::SocketSelector`
- `Sound() : sf::Sound`
- `SoundBuffer() : sf::SoundBuffer`
- `SoundRecorder() : sf::SoundRecorder`
- `SoundSource() : sf::SoundSource`
- `SoundStream() : sf::SoundStream`
- `Sprite() : sf::Sprite`
- `start() : sf::SoundRecorder`
- `stop() : sf::Sound , sf::SoundRecorder , sf::SoundStream`
- `String() : sf::String`
- `substring() : sf::String`

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- t -

- [TcpListener\(\) : sf::TcpListener](#)
- [TcpSocket\(\) : sf::TcpSocket](#)
- [tell\(\) : sf::FileInputStream , sf::InputStream , sf::MemoryInputStream](#)
- [terminate\(\) : sf::Thread](#)
- [Text\(\) : sf::Text](#)
- [Texture\(\) : sf::Texture](#)
- [Thread\(\) : sf::Thread](#)
- [ThreadLocal\(\) : sf::ThreadLocal](#)
- [ThreadLocalPtr\(\) : sf::ThreadLocalPtr< T >](#)
- [Time\(\) : sf::Time](#)
- [toAnsi\(\) : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >](#)
- [toAnsiString\(\) : sf::String](#)
- [toInteger\(\) : sf::Color , sf::IpAddress](#)
- [toLatin1\(\) : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >](#)

- `toString() : sf::IpAddress`
- `toUtf16() : sf::String , sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >`
- `toUtf32() : sf::String , sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >`
- `toUtf8() : sf::String , sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >`
- `toWide() : sf::Utf< 16 > , sf::Utf< 32 > , sf::Utf< 8 >`
- `toWideString() : sf::String`
- `Transform() : sf::Transform`
- `Transformable() : sf::Transformable`
- `transformPoint() : sf::Transform`
- `transformRect() : sf::Transform`
- `TransientContextLock() : sf::GIResource::TransientContextLock`
- `translate() : sf::Transform`

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[**u**](#)[v](#)[w](#)[z](#)[~](#)

- U -

- [UdpSocket\(\) : sf::UdpSocket](#)
- [unbind\(\) : sf::UdpSocket](#)
- [unlock\(\) : sf::Mutex](#)
- [unregisterReader\(\) : sf::SoundFileFactory](#)
- [unregisterWriter\(\) : sf::SoundFileFactory](#)
- [update\(\) : sf::Joystick , sf::Shape , sf::Texture](#)
- [upload\(\) : sf::Ftp](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- V -

- [Vector2\(\) : sf::Vector2< T >](#)
- [Vector3\(\) : sf::Vector3< T >](#)
- [Vertex\(\) : sf::Vertex](#)
- [VertexArray\(\) : sf::VertexArray](#)
- [VideoMode\(\) : sf::VideoMode](#)
- [View\(\) : sf::View](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[**w**](#)[z](#)[~](#)

- W -

- [wait\(\) : sf::SocketSelector , sf::Thread](#)
- [waitEvent\(\) : sf::Window](#)
- [Window\(\) : sf::Window](#)
- [write\(\) : sf::OutputSoundFile , sf::SoundFileWriter](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- Z -

- [zoom\(\) : sf::View](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[**Functions**](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[z](#)[~](#)

- [~AIResource\(\) : sf::AIResource](#)
- [~Context\(\) : sf::Context](#)
- [~Drawable\(\) : sf::Drawable](#)
- [~FileInputStream\(\) : sf::FileInputStream](#)
- [~Font\(\) : sf::Font](#)
- [~Ftp\(\) : sf::Ftp](#)
- [~GIResource\(\) : sf::GIResource](#)
- [~Image\(\) : sf::Image](#)
- [~InputSoundFile\(\) : sf::InputSoundFile](#)
- [~InputStream\(\) : sf::InputStream](#)
- [~Lock\(\) : sf::Lock](#)
- [~Music\(\) : sf::Music](#)
- [~Mutex\(\) : sf::Mutex](#)
- [~OutputSoundFile\(\) : sf::OutputSoundFile](#)

- ~ -

- ~Packet() : sf::Packet
- ~RenderTarget() : sf::RenderTarget
- ~RenderTexture() : sf::RenderTexture
- ~RenderWindow() : sf::RenderWindow
- ~Shader() : sf::Shader
- ~Shape() : sf::Shape
- ~Socket() : sf::Socket
- ~SocketSelector() : sf::SocketSelector
- ~Sound() : sf::Sound
- ~SoundBuffer() : sf::SoundBuffer
- ~SoundBufferRecorder() : sf::SoundBufferRecorder
- ~SoundFileReader() : sf::SoundFileReader
- ~SoundFileWriter() : sf::SoundFileWriter
- ~SoundRecorder() : sf::SoundRecorder
- ~SoundSource() : sf::SoundSource
- ~SoundStream() : sf::SoundStream
- ~Texture() : sf::Texture
- ~Thread() : sf::Thread
- ~ThreadLocal() : sf::ThreadLocal
- ~Transformable() : sf::Transformable
- ~TransientContextLock() : sf::GIResource::TransientContextLock
- ~Window() : sf::Window

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[p](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- a -

- a : sf::Color
- advance : sf::Glyph
- alphaDstFactor : sf::BlendMode
- alphaEquation : sf::BlendMode
- alphaSrcFactor : sf::BlendMode
- alt : sf::Event::KeyEvent
- antialiasingLevel : sf::ContextSettings
- Any : sf::IpAddress
- attributeFlags : sf::ContextSettings
- axis : sf::Event::JoystickMoveEvent

- b -

- b : sf::Color

- bitsPerPixel : sf::VideoMode
- Black : sf::Color
- blendMode : sf::RenderStates
- Blue : sf::Color
- bounds : sf::Glyph
- Broadcast : sf::IpAddress
- button : sf::Event::JoystickButtonEvent , sf::Event::MouseButtonEvent

- C -

- channelCount : sf::SoundFileReader::Info
- code : sf::Event::KeyEvent
- color : sf::Vertex
- colorDstFactor : sf::BlendMode
- colorEquation : sf::BlendMode
- colorSrcFactor : sf::BlendMode
- control : sf::Event::KeyEvent
- CurrentTexture : sf::Shader
- Cyan : sf::Color

- d -

- Default : sf::RenderStates
- delta : sf::Event::MouseWheelEvent , sf::Event::MouseWheelScrollEvent
- depthBits : sf::ContextSettings

- f -

- family : sf::Font::Info
- finger : sf::Event::TouchEvent

- g -

- g : sf::Color
- Green : sf::Color

- h -

- height : sf::Event::SizeEvent , sf::Rect< T > , sf::VideoMode

- i -

- Identity : sf::Transform
- InvalidPos : sf::String

- j -

- joystickButton : sf::Event
- joystickConnect : sf::Event
- joystickId : sf::Event::JoystickButtonEvent , sf::Event::JoystickConnect
sf::Event::JoystickMoveEvent
- joystickMove : sf::Event

- k -

- key : sf::Event

- | -

- left : sf::Rect< T >
- LocalHost : sf::IpAddress

- m -

- m_source : sf::SoundSource
- Magenta : sf::Color
- majorVersion : sf::ContextSettings
- minorVersion : sf::ContextSettings
- mouseButton : sf::Event
- mouseMove : sf::Event
- mouseWheel : sf::Event
- mouseWheelScroll : sf::Event

- n -

- name : sf::Joystick::Identification
- None : sf::IpAddress

- p -

- position : sf::Event::JoystickMoveEvent , sf::Vertex
- productId : sf::Joystick::Identification

- r -

- r : sf::Color

- Red : sf::Color

- S -

- sampleCount : sf::SoundFileReader::Info , sf::SoundStream::Chunk
- sampleRate : sf::SoundFileReader::Info
- samples : sf::SoundStream::Chunk
- sensor : sf::Event
- shader : sf::RenderStates
- shift : sf::Event::KeyEvent
- size : sf::Event
- sRgbCapable : sf::ContextSettings
- stencilBits : sf::ContextSettings
- system : sf::Event::KeyEvent

- t -

- texCoords : sf::Vertex
- text : sf::Event
- texture : sf::RenderStates
- textureRect : sf::Glyph
- top : sf::Rect< T >
- touch : sf::Event
- transform : sf::RenderStates
- Transparent : sf::Color
- type : sf::Event::SensorEvent , sf::Event

- U -

- unicode : sf::Event::TextEvent

- V -

- vendorId : sf::Joystick::Identification

- W -

- wheel : sf::Event::MouseWheelScrollEvent
- White : sf::Color
- width : sf::Event::SizeEvent , sf::Rect< T > , sf::VideoMode

- X -

- x : sf::Event::MouseEvent , sf::Event::MouseMoveEvent , sf::Event::MouseWheelScrollEvent , sf::Event::SensorEvent , sf::Event::Vector3< T >

- Y -

- y : sf::Event::MouseEvent , sf::Event::MouseMoveEvent , sf::Event::MouseWheelScrollEvent , sf::Event::SensorEvent , sf::Event::Vector3< T >
- Yellow : sf::Color

- Z -

- z : sf::Event::SensorEvent , sf::Vector3< T >

■ Zero : sf::Time

SFML is licensed under the terms and conditions of the zlib/png license.
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[**TypeDefs**](#)[Enumerations](#)[Enumerator](#)[Related Functions](#)

- [ConstIterator : sf::String](#)

- [Iterator : sf::String](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[**Enumerations**](#)[Enumerator](#)[Related Functions](#)

- Attribute : [sf::ContextSettings](#)
- Axis : [sf::Joystick](#)
- Button : [sf::Mouse](#)
- CoordinateType : [sf::Texture](#)
- Equation : [sf::BlendMode](#)
- EventType : [sf::Event](#)
- Factor : [sf::BlendMode](#)
- Key : [sf::Keyboard](#)
- Method : [sf::Http::Request](#)
- Status : [sf::Ftp::Response](#) , [sf::Http::Response](#) , [sf::Socket](#) , [sf::Sound](#)
- Style : [sf::Text](#)
- TransferMode : [sf::Ftp](#)
- Type : [sf::Sensor](#) , [sf::Shader](#) , [sf::Socket](#)
- Wheel : [sf::Mouse](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- a -

- A : [sf::Keyboard](#)
- Accelerometer : [sf::Sensor](#)
- Accepted : [sf::Http::Response](#)
- Add : [sf::BlendMode](#) , [sf::Keyboard](#)
- AnyPort : [sf::Socket](#)
- Ascii : [sf::Ftp](#)
- AxisCount : [sf::Joystick](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[**b**](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- b -

- B : [sf::Keyboard](#)
- BackSlash : [sf::Keyboard](#)
- BackSpace : [sf::Keyboard](#)
- BadCommandSequence : [sf::Ftp::Response](#)
- BadGateway : [sf::Http::Response](#)
- BadRequest : [sf::Http::Response](#)
- Binary : [sf::Ftp](#)
- Bold : [sf::Text](#)
- ButtonCount : [sf::Joystick](#) , [sf::Mouse](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- C -

- C : [sf::Keyboard](#)
- Closed : [sf::Event](#)
- ClosingConnection : [sf::Ftp::Response](#)
- ClosingDataConnection : [sf::Ftp::Response](#)
- Comma : [sf::Keyboard](#)
- CommandNotImplemented : [sf::Ftp::Response](#)
- CommandUnknown : [sf::Ftp::Response](#)
- ConnectionClosed : [sf::Ftp::Response](#)
- ConnectionFailed : [sf::Ftp::Response](#) , [sf::Http::Response](#)
- Core : [sf::ContextSettings](#)
- Count : [sf::Event](#) , [sf::Joystick](#) , [sf::Sensor](#)
- Created : [sf::Http::Response](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[**d**](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- d -

- D : [sf::Keyboard](#)
- Dash : [sf::Keyboard](#)
- DataConnectionAlreadyOpened : [sf::Ftp::Response](#)
- DataConnectionOpened : [sf::Ftp::Response](#)
- DataConnectionUnavailable : [sf::Ftp::Response](#)
- Debug : [sf::ContextSettings](#)
- Default : [sf::ContextSettings](#)
- Delete : [sf::Http::Request](#) , [sf::Keyboard](#)
- DirectoryOk : [sf::Ftp::Response](#)
- DirectoryStatus : [sf::Ftp::Response](#)
- Disconnected : [sf::Socket](#)
- Divide : [sf::Keyboard](#)
- Done : [sf::Socket](#)
- Down : [sf::Keyboard](#)

- DstAlpha : sf::BlendMode
 - DstColor : sf::BlendMode
-

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[**e**](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- e -

- E : [sf::Keyboard](#)
- Ebcdic : [sf::Ftp](#)
- End : [sf::Keyboard](#)
- EnteringPassiveMode : [sf::Ftp::Response](#)
- Equal : [sf::Keyboard](#)
- Error : [sf::Socket](#)
- Escape : [sf::Keyboard](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[**f**](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- f -

- [F : sf::Keyboard](#)
- [F1 : sf::Keyboard](#)
- [F10 : sf::Keyboard](#)
- [F11 : sf::Keyboard](#)
- [F12 : sf::Keyboard](#)
- [F13 : sf::Keyboard](#)
- [F14 : sf::Keyboard](#)
- [F15 : sf::Keyboard](#)
- [F2 : sf::Keyboard](#)
- [F3 : sf::Keyboard](#)
- [F4 : sf::Keyboard](#)
- [F5 : sf::Keyboard](#)
- [F6 : sf::Keyboard](#)
- [F7 : sf::Keyboard](#)

- F8 : sf::Keyboard
 - F9 : sf::Keyboard
 - FileActionAborted : sf::Ftp::Response
 - FileActionOk : sf::Ftp::Response
 - FilenameNotAllowed : sf::Ftp::Response
 - FileStatus : sf::Ftp::Response
 - FileUnavailable : sf::Ftp::Response
 - Forbidden : sf::Http::Response
 - Fragment : sf::Shader
-

SFML is licensed under the terms and conditions of the zlib/png
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[**g**](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- g -

- [G : sf::Keyboard](#)
- [GainedFocus : sf::Event](#)
- [GatewayTimeout : sf::Http::Response](#)
- [Geometry : sf::Shader](#)
- [Get : sf::Http::Request](#)
- [Gravity : sf::Sensor](#)
- [Gyroscope : sf::Sensor](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[**h**](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- h -

- H : [sf::Keyboard](#)
- Head : [sf::Http::Request](#)
- HelpMessage : [sf::Ftp::Response](#)
- Home : [sf::Keyboard](#)
- HorizontalWheel : [sf::Mouse](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- i -

- I : [sf::Keyboard](#)
- Insert : [sf::Keyboard](#)
- InsufficientStorageSpace : [sf::Ftp::Response](#)
- InternalServerError : [sf::Http::Response](#)
- InvalidFile : [sf::Ftp::Response](#)
- InvalidResponse : [sf::Ftp::Response](#) , [sf::Http::Response](#)
- Italic : [sf::Text](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[**j**](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- j -

- J : [sf::Keyboard](#)
- JoystickButtonPressed : [sf::Event](#)
- JoystickButtonReleased : [sf::Event](#)
- JoystickConnected : [sf::Event](#)
- JoystickDisconnected : [sf::Event](#)
- JoystickMoved : [sf::Event](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[**k**](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- k -

- K : [sf::Keyboard](#)
- KeyCount : [sf::Keyboard](#)
- KeyPressed : [sf::Event](#)
- KeyReleased : [sf::Event](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- | -

- L : [sf::Keyboard](#)
- LAlt : [sf::Keyboard](#)
- LBracket : [sf::Keyboard](#)
- LControl : [sf::Keyboard](#)
- Left : [sf::Keyboard](#) , [sf::Mouse](#)
- LocalError : [sf::Ftp::Response](#)
- LoggedIn : [sf::Ftp::Response](#)
- LostFocus : [sf::Event](#)
- LShift : [sf::Keyboard](#)
- LSystem : [sf::Keyboard](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[**m**](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- m -

- M : [sf::Keyboard](#)
- Magnetometer : [sf::Sensor](#)
- MaxDatagramSize : [sf::UdpSocket](#)
- Menu : [sf::Keyboard](#)
- Middle : [sf::Mouse](#)
- MouseButtonPressed : [sf::Event](#)
- MouseButtonReleased : [sf::Event](#)
- MouseEntered : [sf::Event](#)
- MouseLeft : [sf::Event](#)
- MouseMoved : [sf::Event](#)
- MouseWheelMoved : [sf::Event](#)
- MouseWheelScrolled : [sf::Event](#)
- MovedPermanently : [sf::Http::Response](#)
- MovedTemporarily : [sf::Http::Response](#)

- `MultipleChoices` : `sf::Http::Response`
 - `Multiply` : `sf::Keyboard`
-

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- n -

- N : [sf::Keyboard](#)
- NeedAccountToLogin : [sf::Ftp::Response](#)
- NeedAccountToStore : [sf::Ftp::Response](#)
- NeedInformation : [sf::Ftp::Response](#)
- NeedPassword : [sf::Ftp::Response](#)
- NoContent : [sf::Http::Response](#)
- Normalized : [sf::Texture](#)
- NotEnoughMemory : [sf::Ftp::Response](#)
- NotFound : [sf::Http::Response](#)
- NotImplemented : [sf::Http::Response](#)
- NotLoggedIn : [sf::Ftp::Response](#)
- NotModified : [sf::Http::Response](#)
- NotReady : [sf::Socket](#)
- Num0 : [sf::Keyboard](#)

- Num1 : sf::Keyboard
- Num2 : sf::Keyboard
- Num3 : sf::Keyboard
- Num4 : sf::Keyboard
- Num5 : sf::Keyboard
- Num6 : sf::Keyboard
- Num7 : sf::Keyboard
- Num8 : sf::Keyboard
- Num9 : sf::Keyboard
- Numpad0 : sf::Keyboard
- Numpad1 : sf::Keyboard
- Numpad2 : sf::Keyboard
- Numpad3 : sf::Keyboard
- Numpad4 : sf::Keyboard
- Numpad5 : sf::Keyboard
- Numpad6 : sf::Keyboard
- Numpad7 : sf::Keyboard
- Numpad8 : sf::Keyboard
- Numpad9 : sf::Keyboard

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- O -

- [O : sf::Keyboard](#)
- [Ok : sf::Ftp::Response , sf::Http::Response](#)
- [One : sf::BlendMode](#)
- [OneMinusDstAlpha : sf::BlendMode](#)
- [OneMinusDstColor : sf::BlendMode](#)
- [OneMinusSrcAlpha : sf::BlendMode](#)
- [OneMinusSrcColor : sf::BlendMode](#)
- [OpeningDataConnection : sf::Ftp::Response](#)
- [Orientation : sf::Sensor](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[**p**](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- p -

- P : [sf::Keyboard](#)
- PageDown : [sf::Keyboard](#)
- PageTypeUnknown : [sf::Ftp::Response](#)
- PageUp : [sf::Keyboard](#)
- ParameterNotImplemented : [sf::Ftp::Response](#)
- ParametersUnknown : [sf::Ftp::Response](#)
- Partial : [sf::Socket](#)
- PartialContent : [sf::Http::Response](#)
- Pause : [sf::Keyboard](#)
- Paused : [sf::SoundSource](#)
- Period : [sf::Keyboard](#)
- Pixels : [sf::Texture](#)
- Playing : [sf::SoundSource](#)
- PointlessCommand : [sf::Ftp::Response](#)

- Post : sf::Http::Request
 - PovX : sf::Joystick
 - PovY : sf::Joystick
 - Put : sf::Http::Request
-

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- q -

- Q : [sf::Keyboard](#)
- Quote : [sf::Keyboard](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- r -

- R : [sf::Joystick](#) , [sf::Keyboard](#)
- RAlt : [sf::Keyboard](#)
- RangeNotSatisfiable : [sf::Http::Response](#)
- RBracket : [sf::Keyboard](#)
- RControl : [sf::Keyboard](#)
- Regular : [sf::Text](#)
- ResetContent : [sf::Http::Response](#)
- Resized : [sf::Event](#)
- RestartMarkerReply : [sf::Ftp::Response](#)
- Return : [sf::Keyboard](#)
- ReverseSubtract : [sf::BlendMode](#)
- Right : [sf::Keyboard](#) , [sf::Mouse](#)
- RShift : [sf::Keyboard](#)
- RSystem : [sf::Keyboard](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[**s**](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- S -

- [S : sf::Keyboard](#)
- [SemiColon : sf::Keyboard](#)
- [SensorChanged : sf::Event](#)
- [ServiceNotAvailable : sf::Http::Response](#)
- [ServiceReady : sf::Ftp::Response](#)
- [ServiceReadySoon : sf::Ftp::Response](#)
- [ServiceUnavailable : sf::Ftp::Response](#)
- [Slash : sf::Keyboard](#)
- [Space : sf::Keyboard](#)
- [SrcAlpha : sf::BlendMode](#)
- [SrcColor : sf::BlendMode](#)
- [Stopped : sf::SoundSource](#)
- [StrikeThrough : sf::Text](#)
- [Subtract : sf::BlendMode , sf::Keyboard](#)

- SystemStatus : sf::Ftp::Response
 - SystemType : sf::Ftp::Response
-

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[**t**](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- t -

- T : [sf::Keyboard](#)
- Tab : [sf::Keyboard](#)
- Tcp : [sf::Socket](#)
- TextEntered : [sf::Event](#)
- Tilde : [sf::Keyboard](#)
- TouchBegan : [sf::Event](#)
- TouchEnded : [sf::Event](#)
- TouchMoved : [sf::Event](#)
- TransferAborted : [sf::Ftp::Response](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[**u**](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- U -

- U : [sf::Joystick](#) , [sf::Keyboard](#)
- Udp : [sf::Socket](#)
- Unauthorized : [sf::Http::Response](#)
- Underlined : [sf::Text](#)
- Unknown : [sf::Keyboard](#)
- Up : [sf::Keyboard](#)
- UserAcceleration : [sf::Sensor](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[**v**](#)[w](#)[x](#)[y](#)[z](#)

- V -

- V : [sf::Joystick](#) , [sf::Keyboard](#)
- VersionNotSupported : [sf::Http::Response](#)
- Vertex : [sf::Shader](#)
- VerticalWheel : [sf::Mouse](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[**w**](#)[x](#)[y](#)[z](#)

- W -

- [W : sf::Keyboard](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[**x**](#)[y](#)[z](#)

- X -

- X : [sf::Joystick](#) , [sf::Keyboard](#)
- XButton1 : [sf::Mouse](#)
- XButton2 : [sf::Mouse](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[**y**](#)[z](#)

- y -

- Y : [sf::Joystick](#) , [sf::Keyboard](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[**Enumerator**](#)[Related Functions](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[j](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)[y](#)[z](#)

- Z -

- [Z : sf::Joystick , sf::Keyboard](#)
- [Zero : sf::BlendMode](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[**Class Members**](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[**Related Functions**](#)

- [operator< : sf::IpAddress](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[**Files**](#)[File List](#)

File List

Here is a list of all documented files with brief descriptions:

 [AIResource.hpp](#)

 [Audio.hpp](#)

 [BlendMode.hpp](#)

 [CircleShape.hpp](#)

 [Clock.hpp](#)

 [Color.hpp](#)

 [Config.hpp](#)

 [Context.hpp](#)

 [ContextSettings.hpp](#)

 [ConvexShape.hpp](#)

 [Drawable.hpp](#)

 [Err.hpp](#)

 [Event.hpp](#)

 [Audio/Export.hpp](#)

 [Graphics/Export.hpp](#)

 **Network/Export.hpp**

 **System/Export.hpp**

 **Window/Export.hpp**

 **FileInputStream.hpp**

 **Font.hpp**

 **Ftp.hpp**

 **GIResource.hpp**

 **GIsI.hpp**

 **Glyph.hpp**

 **Graphics.hpp**

 **Http.hpp**

 **Image.hpp**

 **InputSoundFile.hpp**

 **InputStream.hpp**

 **IpAddress.hpp**

 **Joystick.hpp**

 **Keyboard.hpp**

 **Listener.hpp**

 **Lock.hpp**

 **Main.hpp**

 **mainpage.hpp**

 **MemoryInputStream.hpp**

 **Mouse.hpp**

 **Music.hpp**

 **Mutex.hpp**

 **NativeActivity.hpp**

 **Network.hpp**

 **NonCopyable.hpp**

 **OpenGL.hpp**

 **OutputSoundFile.hpp**

 **Packet.hpp**

 **PrimitiveType.hpp**

 **Rect.hpp**

 **RectangleShape.hpp**

 **RenderStates.hpp**

 **RenderTarget.hpp**

 **RenderTexture.hpp**

 **RenderWindow.hpp**

 **Sensor.hpp**

 **Shader.hpp**

 **Shape.hpp**

 **Sleep.hpp**

 **Socket.hpp**

 **SocketHandle.hpp**

 **SocketSelector.hpp**

 **Sound.hpp**

 **SoundBuffer.hpp**

 **SoundBufferRecorder.hpp**

 **SoundFileFactory.hpp**

 **SoundFileReader.hpp**

 **SoundStreamWriter.hpp**

 **SoundRecorder.hpp**

 **SoundSource.hpp**

 **SoundStream.hpp**

-  **Sprite.hpp**
-  **String.hpp**
-  **System.hpp**
-  **TcpListener.hpp**
-  **TcpSocket.hpp**
-  **Text.hpp**
-  **Texture.hpp**
-  **Thread.hpp**
-  **ThreadLocal.hpp**
-  **ThreadLocalPtr.hpp**
-  **Time.hpp**
-  **Touch.hpp**
-  **Transform.hpp**
-  **Transformable.hpp**
-  **UdpSocket.hpp**
-  **Utf.hpp**
-  **Vector2.hpp**
-  **Vector3.hpp**
-  **Vertex.hpp**
-  **VertexArray.hpp**
-  **VideoMode.hpp**
-  **View.hpp**
-  **Window/Window.hpp**
-  **Window.hpp**
-  **WindowHandle.hpp**
-  **WindowStyle.hpp**

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

AIResource.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_ALRESOURCE_HPP
26 #define SFML_ALRESOURCE_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32
33
34 namespace sf
35 {
```

```
40 class SFML_AUDIO_API AlResource
41 {
42 protected:
43
48     AlResource();
49
54     ~AlResource();
55 };
56
57 } // namespace sf
58
59
60 #endif // SFML_ALRESOURCE_HPP
61
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Audio.hpp

```
1 //  
3 // SFML - Simple and Fast Multimedia Library  
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)  
5 //  
6 // This software is provided 'as-is', without any express or imp  
7 // In no event will the authors be held liable for any damages a  
software.  
8 //  
9 // Permission is granted to anyone to use this software for any  
10 // including commercial applications, and to alter it and redist  
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 n  
17 //  
18 // 2. Altered source versions must be plainly marked as such,  
19 //     and must not be misrepresented as being the original softw  
20 //  
21 // 3. This notice may not be removed or altered from any source  
22 //  
24  
25 #ifndef SFML_AUDIO_HPP  
26 #define SFML_AUDIO_HPP  
27  
29 // Headers  
31  
32 #include <SFML/System.hpp>  
33 #include <SFML/Audio/InputSoundFile.hpp>  
34 #include <SFML/Audio/Listener.hpp>  
35 #include <SFML/Audio/Music.hpp>
```

```
36 #include <SFML/Audio/OutputSoundFile.hpp>
37 #include <SFML/Audio/Sound.hpp>
38 #include <SFML/Audio/SoundBuffer.hpp>
39 #include <SFML/Audio/SoundBufferRecorder.hpp>
40 #include <SFML/Audio/SoundFileFactory.hpp>
41 #include <SFML/Audio/SoundFileReader.hpp>
42 #include <SFML/Audio/SoundFileWriter.hpp>
43 #include <SFML/Audio/SoundRecorder.hpp>
44 #include <SFML/Audio/SoundSource.hpp>
45 #include <SFML/Audio/SoundStream.hpp>
46
47
48 #endif // SFML_AUDIO_HPP
49
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

BlendMode.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_BLENDMODE_HPP
26 #define SFML_BLENDMODE_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32
33
34 namespace sf
35 {
```

```
36
41 struct SFML_GRAPHICS_API BlendMode
42 {
49     enum Factor
50     {
51         Zero,
52         One,
53         SrcColor,
54         OneMinusSrcColor,
55         DstColor,
56         OneMinusDstColor,
57         SrcAlpha,
58         OneMinusSrcAlpha,
59         DstAlpha,
60         OneMinusDstAlpha
61     };
62
69     enum Equation
70     {
71         Add,
72         Subtract,
73         ReverseSubtract
74     };
75
82     BlendMode();
83
95     BlendMode(Factor sourceFactor, Factor destinationFactor, Equ
96
108     BlendMode(Factor colorSourceFactor, Factor colorDestinationF
109             Equation colorBlendEquation, Factor alphaSourceFac
110             Factor alphaDestinationFactor, Equation alphaBlend
111
113     // Member Data
115     Factor colorSrcFactor;
116     Factor colorDstFactor;
117     Equation colorEquation;
118     Factor alphaSrcFactor;
119     Factor alphaDstFactor;
120     Equation alphaEquation;
121 };
122
133 SFML_GRAPHICS_API bool operator ==(const BlendMode& left, const
134
145 SFML_GRAPHICS_API bool operator !=(const BlendMode& left, const
146
148 // Commonly used blending modes
150 SFML_GRAPHICS_API extern const BlendMode BlendAlpha;
151 SFML_GRAPHICS_API extern const BlendMode BlendAdd;
152 SFML_GRAPHICS_API extern const BlendMode BlendMultiply;
153 SFML_GRAPHICS_API extern const BlendMode BlendNone;
154
155 } // namespace sf
```

```
156
157
158 #endif // SFML_BLENDMODE_HPP
159
160
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

CircleShape.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_CIRCLESHAPE_HPP
26 #define SFML_CIRCLESHAPE_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Shape.hpp>
33
34
35 namespace sf
```

```
36  {
41  class SFML_GRAPHICS_API CircleShape : public Shape
42  {
43  public:
44
52      explicit CircleShape(float radius = 0, std::size_t pointCour
53
62      void setRadius(float radius);
63
72      float getRadius() const;
73
82      void setPointCount(std::size_t count);
83
92      virtual std::size_t getPointCount() const;
93
107     virtual Vector2f getPoint(std::size_t index) const;
108
109 private:
110
112     // Member data
114     float m_radius;
115     std::size_t m_pointCount;
116 };
117
118 } // namespace sf
119
120
121 #endif // SFML_CIRCLESHAPE_HPP
122
123
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

Clock.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_CLOCK_HPP
26 #define SFML_CLOCK_HPP
27
29 // Headers
31 #include <SFML/System/Export.hpp>
32 #include <SFML/System/Time.hpp>
33
34
35 namespace sf
```

```
36  {
41  class SFML_SYSTEM_API Clock
42  {
43  public:
44
51      Clock();
52
63      Time getElapsedTime() const;
64
74      Time restart();
75
76  private:
77
79      // Member data
81      Time m_startTime;
82  };
83
84 } // namespace sf
85
86
87 #endif // SFML_CLOCK_HPP
88
89
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Color.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_COLOR_HPP
26 #define SFML_COLOR_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32
33
34 namespace sf
35 {
```

```
40 class SFML_GRAPHICS_API Color
41 {
42 public:
43
44     Color();
45
46     Color(Uint8 red, Uint8 green, Uint8 blue, Uint8 alpha = 255)
47
48     explicit Color(Uint32 color);
49
50     Uint32 toInteger() const;
51
52     // Static member data
53     static const Color Black;
54     static const Color White;
55     static const Color Red;
56     static const Color Green;
57     static const Color Blue;
58     static const Color Yellow;
59     static const Color Magenta;
60     static const Color Cyan;
61     static const Color Transparent;
62
63     // Member data
64     Uint8 r;
65     Uint8 g;
66     Uint8 b;
67     Uint8 a;
68 };
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114     SFML_GRAPHICS_API bool operator ==(const Color& left, const Color& right) const;
115
116     SFML_GRAPHICS_API bool operator !=(const Color& left, const Color& right) const;
117
118     SFML_GRAPHICS_API Color operator +(const Color& left, const Color& right) const;
119
120     SFML_GRAPHICS_API Color operator -(const Color& left, const Color& right) const;
121
122     SFML_GRAPHICS_API Color operator *(const Color& left, const Color& right) const;
123
124     SFML_GRAPHICS_API Color& operator +=(Color& left, const Color& right);
125
126     SFML_GRAPHICS_API Color& operator -=(Color& left, const Color& right);
127
128     SFML_GRAPHICS_API Color& operator *=(Color& left, const Color& right);
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227 } // namespace sf
228
229
230 #endif // SFML_COLOR_HPP
231
232
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Config.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_CONFIG_HPP
26 #define SFML_CONFIG_HPP
27
28
30 // Define the SFML version
32 #define SFML_VERSION_MAJOR 2
33 #define SFML_VERSION_MINOR 4
34 #define SFML_VERSION_PATCH 2
35
```

```
36
38 // Identify the operating system
39 // see
40 // http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compi
41 // rating_system
42
43 #if defined(_WIN32)
44
45 // Windows
46 #define SFML_SYSTEM_WINDOWS
47 #ifndef NOMINMAX
48     #define NOMINMAX
49 #endif
50
51 #elif defined(__APPLE__)
52
53 // Apple platform, see which one it is
54 #include "TargetConditionals.h"
55
56 #if TARGET_OS_IPHONE || TARGET_IPHONE_SIMULATOR
57
58 // iOS
59 #define SFML_SYSTEM_IOS
60
61 #elif TARGET_OS_MAC
62
63 // MacOS
64 #define SFML_SYSTEM_MACOS
65
66 #else
67
68 // Unsupported Apple system
69 #error This Apple operating system is not supported by S
70
71 #endif
72
73 #else
74
75 // Unix system, see which one it is
76 #if defined(__ANDROID__)
77
78 // Android
79 #define SFML_SYSTEM_ANDROID
80
81 #elif defined(__linux__)
82
83 // Linux
84 #define SFML_SYSTEM_LINUX
85
86 #elif defined(__FreeBSD__) || defined(__FreeBSD_kernel__)
87
88 // FreeBSD
89 #define SFML_SYSTEM_FREEBSD
```

```
88
89     #else
90
91         // Unsupported UNIX system
92         #error This UNIX operating system is not supported by SF
93
94     #endif
95
96 #else
97
98     // Unsupported system
99     #error This operating system is not supported by SFML librar
100
101 #endif
102
103
104 // Define a portable debug macro
105 #if !defined(NDEBUG)
106
107     #define SFML_DEBUG
108
109 #endif
110
111 #endif
112
113
114 // Define helpers to create portable import / export macros for
115 #if !defined(SFML_STATIC)
116
117     #if defined(SFML_SYSTEM_WINDOWS)
118
119         // Windows compilers need specific (and different) keywo
120         #define SFML_API_EXPORT __declspec(dllexport)
121         #define SFML_API_IMPORT __declspec(dllimport)
122
123         // For Visual C++ compilers, we also need to turn off th
124         #ifdef _MSC_VER
125
126             #pragma warning(disable: 4251)
127
128         #endif
129
130     #endif
131
132 #else // Linux, FreeBSD, Mac OS X
133
134     #if __GNUC__ >= 4
135
136         // GCC 4 has special keywords for showing/hidding sy
137         // the same keyword is used for both importing and e
138         #define SFML_API_EXPORT __attribute__((visibility
139         #define SFML_API_IMPORT __attribute__((visibility
140
141     #else
142
143         // GCC < 4 has no mechanism to explicitly hide symb
```

```
144         #define SFML_API_EXPORT
145         #define SFML_API_IMPORT
146
147     #endif
148
149 #endif
150
151 #else
152
153     // Static build doesn't need import/export macros
154     #define SFML_API_EXPORT
155     #define SFML_API_IMPORT
156
157 #endif
158
159
160 // Cross-platform warning for deprecated functions and classes
161 //
162 // Usage:
163 // class SFML_DEPRECATED MyClass
164 // {
165 //     SFML_DEPRECATED void memberFunc();
166 // };
167 //
168 // SFML_DEPRECATED void globalFunc();
169 #if defined(SFML_NO_DEPRECATED_WARNINGS)
170
171     // User explicitly requests to disable deprecation warnings
172     #define SFML_DEPRECATED
173
174 #elif defined(_MSC_VER)
175
176     // Microsoft C++ compiler
177     // Note: On newer MSVC versions, using deprecated functions
178     // order to
179     // trigger a warning instead of an error, the compiler flag
180     // specified.
181     #define SFML_DEPRECATED __declspec(deprecated)
182
183 #elif defined(__GNUC__)
184
185     // g++ and Clang
186     #define SFML_DEPRECATED __attribute__ ((deprecated))
187
188 #else
189
190     // Other compilers are not supported, leave class or function
191     // With a bit of luck, the #pragma directive works, otherwise
192     // error!) for unrecognized #pragma.
193     #pragma message("SFML_DEPRECATED is not supported for your c
194     SFML team")
195     #define SFML_DEPRECATED
```

```

194
195 #endif
196
197
199 // Define portable fixed-size types
201 namespace sf
202 {
203     // All "common" platforms use the same size for char, short
204     // (basically there are 3 types for 3 sizes, so no other mat
205     // we can use them without doing any kind of check
206
207     // 8 bits integer types
208     typedef signed char Int8;
209     typedef unsigned char Uint8;
210
211     // 16 bits integer types
212     typedef signed short Int16;
213     typedef unsigned short Uint16;
214
215     // 32 bits integer types
216     typedef signed int Int32;
217     typedef unsigned int Uint32;
218
219     // 64 bits integer types
220     #if defined(_MSC_VER)
221         typedef signed __int64 Int64;
222         typedef unsigned __int64 Uint64;
223     #else
224         typedef signed long long Int64;
225         typedef unsigned long long Uint64;
226     #endif
227
228 } // namespace sf
229
230
231 #endif // SFML_CONFIG_HPP

```

Context.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_CONTEXT_HPP
26 #define SFML_CONTEXT_HPP
27
29 // Headers
31 #include <SFML/Window/Export.hpp>
32 #include <SFML/Window/GlResource.hpp>
33 #include <SFML/Window/ContextSettings.hpp>
34 #include <SFML/System/NonCopyable.hpp>
35
```

```

36
37 namespace sf
38 {
39 namespace priv
40 {
41     class GlContext;
42 }
43
44 typedef void (*GlFunctionPointer)();
45
50 class SFML_WINDOW_API Context : GlResource, NonCopyable
51 {
52 public:
53
60     Context();
61
68     ~Context();
69
78     bool setActive(bool active);
79
90     const ContextSettings& getSettings() const;
91
100    static bool isExtensionAvailable(const char* name);
101
110    static GlFunctionPointer getFunction(const char* name);
111
118    static const Context* getActiveContext();
119
131    Context(const ContextSettings& settings, unsigned int width,
132
133 private:
134
136     // Member data
138     priv::GlContext* m_context;
139 }
140
141 } // namespace sf
142
143
144 #endif // SFML_CONTEXT_HPP
145

```

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

ContextSettings.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_CONTEXTSETTINGS_HPP
26 #define SFML_CONTEXTSETTINGS_HPP
27
28
29 namespace sf
30 {
36 struct ContextSettings
37 {
42     enum Attribute
```

```

43     {
44         Default = 0,
45         Core    = 1 << 0,
46         Debug   = 1 << 2
47     };
48
61     explicit ContextSettings(unsigned int depth = 0, unsigned int
antialiasing = 0, unsigned int major = 1, unsigned int minor = 1,
Default, bool sRgb = false) :
62         depthBits      (depth),
63         stencilBits    (stencil),
64         antialiasingLevel(antialiasing),
65         majorVersion   (major),
66         minorVersion   (minor),
67         attributeFlags (attributes),
68         sRgbCapable    (sRgb)
69     {
70     }
71
73     // Member data
75     unsigned int depthBits;
76     unsigned int stencilBits;
77     unsigned int antialiasingLevel;
78     unsigned int majorVersion;
79     unsigned int minorVersion;
80     Uint32       attributeFlags;
81     bool          sRgbCapable;
82 };
83
84 } // namespace sf
85
86
87 #endif // SFML_CONTEXTSETTINGS_HPP
88
89

```

ConvexShape.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_CONVEXSHAPE_HPP
26 #define SFML_CONVEXSHAPE_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Shape.hpp>
33 #include <vector>
34
35
```

```
36 namespace sf
37 {
42 class SFML_GRAPHICS_API ConvexShape : public Shape
43 {
44 public:
45
52     explicit ConvexShape(std::size_t pointCount = 0);
53
64     void setPointCount(std::size_t count);
65
74     virtual std::size_t getPointCount() const;
75
91     void setPoint(std::size_t index, const Vector2f& point);
92
108    virtual Vector2f getPoint(std::size_t index) const;
109
110 private:
111
113     // Member data
115     std::vector<Vector2f> m_points;
116 };
117
118 } // namespace sf
119
120
121 #endif // SFML_CONVEXSHAPE_HPP
122
123
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Drawable.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_DRAWABLE_HPP
26 #define SFML_DRAWABLE_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/RenderStates.hpp>
33
34
35 namespace sf
```

```
36 {
37 class RenderTarget;
38
39 class SFML_GRAPHICS_API Drawable
40 {
41 public:
42     virtual ~Drawable() {}
43
44 protected:
45     friend class RenderTarget;
46
47     virtual void draw(RenderTarget& target, RenderStates states)
48 };
49
50 } // namespace sf
51
52 #endif // SFML_DRAWABLE_HPP
53
54
55
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Err.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_ERR_HPP
26 #define SFML_ERR_HPP
27
29 // Headers
31 #include <SFML/System/Export.hpp>
32 #include <iostream>
33
34
35 namespace sf
```

```
36 {
41 SFML_SYSTEM_API std::ostream& err();
42
43 } // namespace sf
44
45
46 #endif // SFML_ERR_HPP
47
48
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

Event.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_EVENT_HPP
26 #define SFML_EVENT_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32 #include <SFML/Window/Joystick.hpp>
33 #include <SFML/Window/Keyboard.hpp>
34 #include <SFML/Window/Mouse.hpp>
35 #include <SFML/Window/Sensor.hpp>
```

```
36
37
38 namespace sf
39 {
40     class Event
41     {
42         public:
43
44             struct SizeEvent
45             {
46                 unsigned int width;
47                 unsigned int height;
48             };
49
50             struct KeyEvent
51             {
52                 Keyboard::Key code;
53                 bool alt;
54                 bool control;
55                 bool shift;
56                 bool system;
57             };
58
59             struct TextEvent
60             {
61                 Uint32 unicode;
62             };
63
64             struct MouseMoveEvent
65             {
66                 int x;
67                 int y;
68             };
69
70             struct MouseButtonEvent
71             {
72                 Mouse::Button button;
73                 int x;
74                 int y;
75             };
76
77             struct MouseWheelEvent
78             {
79                 int delta;
80                 int x;
81                 int y;
82             };
83
84             struct MouseWheelScrollView
85             {
86                 Mouse::Wheel wheel;
87                 float delta;
88             };
89
90             struct MouseWheelEvent
91             {
92                 int delta;
93                 int x;
94                 int y;
95             };
96
97             struct MouseWheelScrollView
98             {
99                 Mouse::Wheel wheel;
100                float delta;
101            };
102
103            struct MouseWheelEvent
104            {
105                int delta;
106                int x;
107                int y;
108            };
109
110            struct MouseWheelScrollView
111            {
112                Mouse::Wheel wheel;
113                float delta;
114            };
115
116            struct MouseWheelEvent
117            {
118                int delta;
119                int x;
120                int y;
121            };
122
123            struct MouseWheelScrollView
124            {
125                Mouse::Wheel wheel;
126                float delta;
127            };
128
129            struct MouseWheelEvent
130            {
131                int delta;
132                int x;
133                int y;
134            };
135
136            struct MouseWheelScrollView
137            {
138                Mouse::Wheel wheel;
139                float delta;
140            };
141
142            struct MouseWheelEvent
143            {
144                int delta;
145                int x;
146                int y;
147            };
148
149            struct MouseWheelScrollView
150            {
151                Mouse::Wheel wheel;
152                float delta;
153            };
154
155            struct MouseWheelEvent
156            {
157                int delta;
158                int x;
159                int y;
160            };
161
162            struct MouseWheelScrollView
163            {
164                Mouse::Wheel wheel;
165                float delta;
166            };
167
168            struct MouseWheelEvent
169            {
170                int delta;
171                int x;
172                int y;
173            };
174
175            struct MouseWheelScrollView
176            {
177                Mouse::Wheel wheel;
178                float delta;
179            };
180
181            struct MouseWheelEvent
182            {
183                int delta;
184                int x;
185                int y;
186            };
187
188            struct MouseWheelScrollView
189            {
189                Mouse::Wheel wheel;
190                float delta;
191            };
192
193            struct MouseWheelEvent
194            {
195                int delta;
196                int x;
197                int y;
198            };
199
200            struct MouseWheelScrollView
201            {
202                Mouse::Wheel wheel;
203                float delta;
204            };
205
206            struct MouseWheelEvent
207            {
208                int delta;
209                int x;
210                int y;
211            };
212
213            struct MouseWheelScrollView
214            {
215                Mouse::Wheel wheel;
216                float delta;
217            };
218
219            struct MouseWheelEvent
220            {
221                int delta;
222                int x;
223                int y;
224            };
225
226            struct MouseWheelScrollView
227            {
228                Mouse::Wheel wheel;
229                float delta;
230            };
231
232            struct MouseWheelEvent
233            {
234                int delta;
235                int x;
236                int y;
237            };
238
239            struct MouseWheelScrollView
239            {
240                Mouse::Wheel wheel;
241                float delta;
242            };
243
244            struct MouseWheelEvent
245            {
246                int delta;
247                int x;
248                int y;
249            };
250
251            struct MouseWheelScrollView
252            {
253                Mouse::Wheel wheel;
254                float delta;
255            };
256
257            struct MouseWheelEvent
258            {
259                int delta;
260                int x;
261                int y;
262            };
263
264            struct MouseWheelScrollView
265            {
266                Mouse::Wheel wheel;
267                float delta;
268            };
269
270            struct MouseWheelEvent
271            {
272                int delta;
273                int x;
274                int y;
275            };
276
277            struct MouseWheelScrollView
278            {
279                Mouse::Wheel wheel;
280                float delta;
281            };
282
283            struct MouseWheelEvent
284            {
285                int delta;
286                int x;
287                int y;
288            };
289
290            struct MouseWheelScrollView
291            {
292                Mouse::Wheel wheel;
293                float delta;
294            };
295
296            struct MouseWheelEvent
297            {
298                int delta;
299                int x;
300                int y;
301            };
302
303            struct MouseWheelScrollView
304            {
305                Mouse::Wheel wheel;
306                float delta;
307            };
308
309            struct MouseWheelEvent
310            {
311                int delta;
312                int x;
313                int y;
314            };
315
316            struct MouseWheelScrollView
317            {
318                Mouse::Wheel wheel;
319                float delta;
320            };
321
322            struct MouseWheelEvent
323            {
324                int delta;
325                int x;
326                int y;
327            };
328
329            struct MouseWheelScrollView
329            {
330                Mouse::Wheel wheel;
331                float delta;
332            };
333
334            struct MouseWheelEvent
335            {
336                int delta;
337                int x;
338                int y;
339            };
340
341            struct MouseWheelScrollView
342            {
343                Mouse::Wheel wheel;
344                float delta;
345            };
346
347            struct MouseWheelEvent
348            {
349                int delta;
350                int x;
351                int y;
352            };
353
354            struct MouseWheelScrollView
355            {
356                Mouse::Wheel wheel;
357                float delta;
358            };
359
360            struct MouseWheelEvent
361            {
362                int delta;
363                int x;
364                int y;
365            };
366
367            struct MouseWheelScrollView
368            {
369                Mouse::Wheel wheel;
370                float delta;
371            };
372
373            struct MouseWheelEvent
374            {
375                int delta;
376                int x;
377                int y;
378            };
379
380            struct MouseWheelScrollView
381            {
382                Mouse::Wheel wheel;
383                float delta;
384            };
385
386            struct MouseWheelEvent
387            {
388                int delta;
389                int x;
390                int y;
391            };
392
393            struct MouseWheelScrollView
394            {
395                Mouse::Wheel wheel;
396                float delta;
397            };
398
399            struct MouseWheelEvent
400            {
401                int delta;
402                int x;
403                int y;
404            };
405
406            struct MouseWheelScrollView
407            {
408                Mouse::Wheel wheel;
409                float delta;
410            };
411
412            struct MouseWheelEvent
413            {
414                int delta;
415                int x;
416                int y;
417            };
418
419            struct MouseWheelScrollView
419            {
420                Mouse::Wheel wheel;
421                float delta;
422            };
423
424            struct MouseWheelEvent
425            {
426                int delta;
427                int x;
428                int y;
429            };
430
431            struct MouseWheelScrollView
432            {
433                Mouse::Wheel wheel;
434                float delta;
435            };
436
437            struct MouseWheelEvent
438            {
439                int delta;
440                int x;
441                int y;
442            };
443
444            struct MouseWheelScrollView
445            {
446                Mouse::Wheel wheel;
447                float delta;
448            };
449
450            struct MouseWheelEvent
451            {
452                int delta;
453                int x;
454                int y;
455            };
456
457            struct MouseWheelScrollView
458            {
459                Mouse::Wheel wheel;
460                float delta;
461            };
462
463            struct MouseWheelEvent
464            {
465                int delta;
466                int x;
467                int y;
468            };
469
470            struct MouseWheelScrollView
471            {
472                Mouse::Wheel wheel;
473                float delta;
474            };
475
476            struct MouseWheelEvent
477            {
478                int delta;
479                int x;
480                int y;
481            };
482
483            struct MouseWheelScrollView
484            {
485                Mouse::Wheel wheel;
486                float delta;
487            };
488
489            struct MouseWheelEvent
490            {
491                int delta;
492                int x;
493                int y;
494            };
495
496            struct MouseWheelScrollView
497            {
498                Mouse::Wheel wheel;
499                float delta;
500            };
501
502            struct MouseWheelEvent
503            {
504                int delta;
505                int x;
506                int y;
507            };
508
509            struct MouseWheelScrollView
509            {
510                Mouse::Wheel wheel;
511                float delta;
512            };
513
514            struct MouseWheelEvent
515            {
516                int delta;
517                int x;
518                int y;
519            };
520
521            struct MouseWheelScrollView
522            {
523                Mouse::Wheel wheel;
524                float delta;
525            };
526
527            struct MouseWheelEvent
528            {
529                int delta;
530                int x;
531                int y;
532            };
533
534            struct MouseWheelScrollView
535            {
536                Mouse::Wheel wheel;
537                float delta;
538            };
539
540            struct MouseWheelEvent
541            {
542                int delta;
543                int x;
544                int y;
545            };
546
547            struct MouseWheelScrollView
548            {
549                Mouse::Wheel wheel;
550                float delta;
551            };
552
553            struct MouseWheelEvent
554            {
555                int delta;
556                int x;
557                int y;
558            };
559
560            struct MouseWheelScrollView
561            {
562                Mouse::Wheel wheel;
563                float delta;
564            };
565
566            struct MouseWheelEvent
567            {
568                int delta;
569                int x;
570                int y;
571            };
572
573            struct MouseWheelScrollView
574            {
575                Mouse::Wheel wheel;
576                float delta;
577            };
578
579            struct MouseWheelEvent
580            {
581                int delta;
582                int x;
583                int y;
584            };
585
586            struct MouseWheelScrollView
587            {
588                Mouse::Wheel wheel;
589                float delta;
590            };
591
592            struct MouseWheelEvent
593            {
594                int delta;
595                int x;
596                int y;
597            };
598
599            struct MouseWheelScrollView
599            {
600                Mouse::Wheel wheel;
601                float delta;
602            };
603
604            struct MouseWheelEvent
605            {
606                int delta;
607                int x;
608                int y;
609            };
610
611            struct MouseWheelScrollView
612            {
613                Mouse::Wheel wheel;
614                float delta;
615            };
616
617            struct MouseWheelEvent
618            {
619                int delta;
620                int x;
621                int y;
622            };
623
624            struct MouseWheelScrollView
625            {
626                Mouse::Wheel wheel;
627                float delta;
628            };
629
630            struct MouseWheelEvent
631            {
632                int delta;
633                int x;
634                int y;
635            };
636
637            struct MouseWheelScrollView
638            {
639                Mouse::Wheel wheel;
640                float delta;
641            };
642
643            struct MouseWheelEvent
644            {
645                int delta;
646                int x;
647                int y;
648            };
649
650            struct MouseWheelScrollView
651            {
652                Mouse::Wheel wheel;
653                float delta;
654            };
655
656            struct MouseWheelEvent
657            {
658                int delta;
659                int x;
660                int y;
661            };
662
663            struct MouseWheelScrollView
664            {
665                Mouse::Wheel wheel;
666                float delta;
667            };
668
669            struct MouseWheelEvent
670            {
671                int delta;
672                int x;
673                int y;
674            };
675
676            struct MouseWheelScrollView
677            {
678                Mouse::Wheel wheel;
679                float delta;
680            };
681
682            struct MouseWheelEvent
683            {
684                int delta;
685                int x;
686                int y;
687            };
688
689            struct MouseWheelScrollView
689            {
690                Mouse::Wheel wheel;
691                float delta;
692            };
693
694            struct MouseWheelEvent
695            {
696                int delta;
697                int x;
698                int y;
699            };
699
699            struct MouseWheelScrollView
700            {
701                Mouse::Wheel wheel;
702                float delta;
703            };
704
705            struct MouseWheelEvent
706            {
707                int delta;
708                int x;
709                int y;
710            };
711
712            struct MouseWheelScrollView
713            {
714                Mouse::Wheel wheel;
715                float delta;
716            };
717
718            struct MouseWheelEvent
719            {
720                int delta;
721                int x;
722                int y;
723            };
724
725            struct MouseWheelScrollView
726            {
727                Mouse::Wheel wheel;
728                float delta;
729            };
730
731            struct MouseWheelEvent
732            {
733                int delta;
734                int x;
735                int y;
736            };
737
738            struct MouseWheelScrollView
739            {
739                Mouse::Wheel wheel;
740                float delta;
741            };
742
743            struct MouseWheelEvent
744            {
745                int delta;
746                int x;
747                int y;
748            };
749
749            struct MouseWheelScrollView
750            {
751                Mouse::Wheel wheel;
752                float delta;
753            };
754
755            struct MouseWheelEvent
756            {
757                int delta;
758                int x;
759                int y;
760            };
761
762            struct MouseWheelScrollView
763            {
764                Mouse::Wheel wheel;
765                float delta;
766            };
767
768            struct MouseWheelEvent
769            {
770                int delta;
771                int x;
772                int y;
773            };
774
775            struct MouseWheelScrollView
776            {
777                Mouse::Wheel wheel;
778                float delta;
779            };
780
781            struct MouseWheelEvent
782            {
783                int delta;
784                int x;
785                int y;
786            };
787
787            struct MouseWheelScrollView
788            {
789                Mouse::Wheel wheel;
789                float delta;
790            };
791
792            struct MouseWheelEvent
793            {
794                int delta;
795                int x;
796                int y;
797            };
798
798            struct MouseWheelScrollView
799            {
799                Mouse::Wheel wheel;
800                float delta;
801            };
802
803            struct MouseWheelEvent
804            {
805                int delta;
806                int x;
807                int y;
808            };
809
809            struct MouseWheelScrollView
810            {
811                Mouse::Wheel wheel;
812                float delta;
813            };
814
815            struct MouseWheelEvent
816            {
817                int delta;
818                int x;
819                int y;
820            };
821
821            struct MouseWheelScrollView
822            {
823                Mouse::Wheel wheel;
824                float delta;
825            };
826
827            struct MouseWheelEvent
828            {
829                int delta;
830                int x;
831                int y;
832            };
833
833            struct MouseWheelScrollView
834            {
835                Mouse::Wheel wheel;
836                float delta;
837            };
838
839            struct MouseWheelEvent
840            {
841                int delta;
842                int x;
843                int y;
844            };
845
845            struct MouseWheelScrollView
846            {
847                Mouse::Wheel wheel;
848                float delta;
849            };
850
851            struct MouseWheelEvent
852            {
853                int delta;
854                int x;
855                int y;
856            };
857
857            struct MouseWheelScrollView
858            {
859                Mouse::Wheel wheel;
860                float delta;
861            };
862
863            struct MouseWheelEvent
864            {
865                int delta;
866                int x;
867                int y;
868            };
869
869            struct MouseWheelScrollView
870            {
871                Mouse::Wheel wheel;
872                float delta;
873            };
874
875            struct MouseWheelEvent
876            {
877                int delta;
878                int x;
879                int y;
880            };
881
881            struct MouseWheelScrollView
882            {
883                Mouse::Wheel wheel;
884                float delta;
885            };
886
887            struct MouseWheelEvent
888            {
889                int delta;
889                int x;
890                int y;
891            };
892
892            struct MouseWheelScrollView
893            {
893                Mouse::Wheel wheel;
894                float delta;
895            };
896
897            struct MouseWheelEvent
898            {
899                int delta;
900                int x;
901                int y;
902            };
903
903            struct MouseWheelScrollView
904            {
905                Mouse::Wheel wheel;
906                float delta;
907            };
908
909            struct MouseWheelEvent
910            {
911                int delta;
912                int x;
913                int y;
914            };
915
915            struct MouseWheelScrollView
916            {
917                Mouse::Wheel wheel;
918                float delta;
919            };
920
921            struct MouseWheelEvent
922            {
923                int delta;
924                int x;
925                int y;
926            };
927
927            struct MouseWheelScrollView
928            {
929                Mouse::Wheel wheel;
929                float delta;
930            };
931
932            struct MouseWheelEvent
933            {
934                int delta;
935                int x;
936                int y;
937            };
938
938            struct MouseWheelScrollView
939            {
939                Mouse::Wheel wheel;
940                float delta;
941            };
942
943            struct MouseWheelEvent
944            {
945                int delta;
946                int x;
947                int y;
948            };
949
949            struct MouseWheelScrollView
950            {
951                Mouse::Wheel wheel;
952                float delta;
953            };
954
955            struct MouseWheelEvent
956            {
957                int delta;
958                int x;
959                int y;
960            };
961
961            struct MouseWheelScrollView
962            {
963                Mouse::Wheel wheel;
964                float delta;
965            };
966
967            struct MouseWheelEvent
968            {
969                int delta;
970                int x;
971                int y;
972            };
973
973            struct MouseWheelScrollView
974            {
974                Mouse::Wheel wheel;
975                float delta;
976            };
977
978            struct MouseWheelEvent
979            {
979                int delta;
980                int x;
981                int y;
982            };
983
983            struct MouseWheelScrollView
984            {
984                Mouse::Wheel wheel;
985                float delta;
986            };
987
988            struct MouseWheelEvent
989            {
989                int delta;
990                int x;
991                int y;
992            };
993
993            struct MouseWheelScrollView
994            {
994                Mouse::Wheel wheel;
995                float delta;
996            };
997
998            struct MouseWheelEvent
999            {
999                int delta;
1000               int x;
1001              int y;
1002            };
1003
1003            struct MouseWheelScrollView
1004            {
1004                Mouse::Wheel wheel;
1005                float delta;
1006            };
1007
1008            struct MouseWheelEvent
1009            {
1009                int delta;
1010               int x;
1011              int y;
1012            };
1013
1013            struct MouseWheelScrollView
1014            {
1014                Mouse::Wheel wheel;
1015                float delta;
1016            };
1017
1018            struct MouseWheelEvent
1019            {
1019                int delta;
1020               int x;
1021              int y;
1022            };
1023
1023            struct MouseWheelScrollView
1024            {
1024                Mouse::Wheel wheel;
1025                float delta;
1026            };
1027
1028            struct MouseWheelEvent
1029            {
1029                int delta;
1030               int x;
1031              int y;
1032            };
1033
1033            struct MouseWheelScrollView
1034            {
1034                Mouse::Wheel wheel;
1035                float delta;
1036            };
1037
1038            struct MouseWheelEvent
1039            {
1039                int delta;
1040               int x;
1041              int y;
1042            };
1043
1043            struct MouseWheelScrollView
1044            {
1044                Mouse::Wheel wheel;
1045                float delta;
1046            };
1047
1048            struct MouseWheelEvent
1049            {
1049                int delta;
1050               int x;
1051              int y;
1052            };
1053
1053            struct MouseWheelScrollView
1054            {
1054                Mouse::Wheel wheel;
1055                float delta;
1056            };
1057
1058            struct MouseWheelEvent
1059            {
1059                int delta;
1060               int x;
1061              int y;
1062            };
1063
1063            struct MouseWheelScrollView
1064            {
1064                Mouse::Wheel wheel;
1065                float delta;
1066            };
1067
1068            struct MouseWheelEvent
1069            {
1069                int delta;
1070               int x;
1071              int y;
1072            };
1073
1073            struct MouseWheelScrollView
1074            {
1074                Mouse::Wheel wheel;
1075                float delta;
1076            };
1077
1078            struct MouseWheelEvent
1079            {
1079                int delta;
1080               int x;
1081              int y;
1082            };
1083
1083            struct MouseWheelScrollView
1084            {
1084                Mouse::Wheel wheel;
1085                float delta;
1086            };
1087
1088            struct MouseWheelEvent
1089            {
1089                int delta;
1090               int x;
1091              int y;
1092            };
1093
1093            struct MouseWheelScrollView
1094            {
1094                Mouse::Wheel wheel;
1095                float delta;
1096            };
1097
1098            struct MouseWheelEvent
1099            {
1099                int delta;
1100               int x;
1101              int y;
1102            };
1103
1103            struct MouseWheelScrollView
1104            {
1104                Mouse::Wheel wheel;
1105                float delta;
1106            };
1107
1108            struct MouseWheelEvent
1109            {
1109                int delta;
1110               int x;
1111              int y;
1112            };
1113
1113            struct MouseWheelScrollView
1114            {
1114                Mouse::Wheel wheel;
1115                float delta;
1116            };
1117
1118            struct MouseWheelEvent
1119            {
1119                int delta;
1120               int x;
1121              int y;
1122            };
1123
1123            struct MouseWheelScrollView
1124            {
1124                Mouse::Wheel wheel;
1125                float delta;
1126            };
1127
1128            struct MouseWheelEvent
1129            {
1129                int delta;
1130               int x;
1131              int y;
1132            };
1133
1133            struct MouseWheelScrollView
1134            {
1134                Mouse::Wheel wheel;
1135                float delta;
1136            };
1137
1138            struct MouseWheelEvent
1139            {
1139                int delta;
1140               int x;
1141              int y;
1142            };
1143
1143            struct MouseWheelScrollView
1144            {
1144                Mouse::Wheel wheel;
1145                float delta;
1146            };
1147
1148            struct MouseWheelEvent
1149            {
1149                int delta;
1150               int x;
1151              int y;
1152            };
1153
1153            struct MouseWheelScrollView
1154            {
1154                Mouse::Wheel wheel;
1155                float delta;
1156            };
1157
1158            struct MouseWheelEvent
1159            {
1159                int delta;
1160               int x;
1161              int y;
1162            };
1163
1163            struct MouseWheelScrollView
1164            {
1164                Mouse::Wheel wheel;
1165                float delta;
1166            };
1167
1168            struct MouseWheelEvent
1169            {
1169                int delta;
1170               int x;
1171              int y;
1172            };
1173
1173            struct MouseWheelScrollView
1174            {
1174                Mouse::Wheel wheel;
1175                float delta;
1176            };
1177
1178            struct MouseWheelEvent
1179            {
1179                int delta;
1180               int x;
1181              int y;
1182            };
1183
1183            struct MouseWheelScrollView
1184            {
1184                Mouse::Wheel wheel;
1185                float delta;
1186            };
1187
1188            struct MouseWheelEvent
1189            {
1189                int delta;
1190               int x;
1191              int y;
1192            };
1193
1193            struct MouseWheelScrollView
1194            {
1194                Mouse::Wheel wheel;
1195                float delta;
1196            };
1197
1198            struct MouseWheelEvent
1199            {
1199                int delta;
1200               int x;
1201              int y;
1202            };
1203
1203            struct MouseWheelScrollView
1204            {
1204                Mouse::Wheel wheel;
1205                float delta;
1206            };
1207
1208            struct MouseWheelEvent
1209            {
1209                int delta;
1210               int x;
1211              int y;
1212            };
1213
1213            struct MouseWheelScrollView
1214            {
1214                Mouse::Wheel wheel;
1215                float delta;
1216            };
1217
1218            struct MouseWheelEvent
1219            {
1219                int delta;
1220               int x;
1221              int y;
1222            };
1223
1223            struct MouseWheelScrollView
1224            {
1224                Mouse::Wheel wheel;
1225                float delta;
1226            };
1227
1228            struct MouseWheelEvent
1229            {
1229                int delta;
1230               int x;
1231              int y;
1232            };
1233
1233            struct MouseWheelScrollView
1234            {
1234                Mouse::Wheel wheel;
1235                float delta;
1236            };
1237
1238            struct MouseWheelEvent
1239            {
1239                int delta;
1240               int x;
1241              int y;
1242            };
1243
1243            struct MouseWheelScrollView
1244            {
1244                Mouse::Wheel wheel;
1245                float delta;
1246            };
1247
1248            struct MouseWheelEvent
1249            {
1249                int delta;
1250               int x;
1251              int y;
1252            };
1253
1253            struct MouseWheelScrollView
1254            {
1254                Mouse::Wheel wheel;
1255                float delta;
1256            };
1257
1258            struct MouseWheelEvent
1259            {
1259                int delta;
1260               int x;
1261              int y;
1262            };
1263
1263            struct MouseWheelScrollView
1264            {
1264                Mouse::Wheel wheel;
1265                float delta;
1266            };
1267
1268            struct MouseWheelEvent
1269            {
1269                int delta;
1270               int x;
1271              int y;
1272            };
1273
1273            struct MouseWheelScrollView
1274            {
1274                Mouse::Wheel wheel;
1275                float delta;
1276            };
1277
1278            struct MouseWheelEvent
1279            {
1279                int delta;
1280               int x;
1281              int y;
1282            };
1283
1283            struct MouseWheelScrollView
1284            {
1284                Mouse::Wheel wheel;
1285                float delta;
1286            };
1287
1288            struct MouseWheelEvent
1289            {
1289                int delta;
1290               int x;
1291              int y;
1292            };
1293
1293            struct MouseWheelScrollView
1294            {
1294                Mouse::Wheel wheel;
1295                float delta;
1296            };
1297
1298            struct MouseWheelEvent
1299            {
1299                int delta;
1300               int x;
1301              int y;
1302            };
1303
1303            struct MouseWheelScrollView
1304            {
1304                Mouse::Wheel wheel;
1305                float delta;
1306            };
1307
1308            struct MouseWheelEvent
1309            {
1309                int delta;
1310               int x;
1311              int y;
1312            };
1313
1313            struct MouseWheelScrollView
1314            {
1314                Mouse::Wheel wheel;
1315                float delta;
1316            };
1317
1318            struct MouseWheelEvent
1319            {
1319                int delta;
1320               int x;
1321              int y;
1322            };
1323
1323            struct MouseWheelScrollView
1324            {
1324                Mouse::Wheel wheel;
1325                float delta;
1326            };
1327
1328            struct MouseWheelEvent
1329            {
1329                int delta;
1330               int x;
1331              int y;
1332            };
1333
1333            struct MouseWheelScrollView
1334            {
1334                Mouse::Wheel wheel;
1335                float delta;
1336            };
1337
1338            struct MouseWheelEvent
1339            {
1339                int delta;
1340               int x;
1341              int y;
1342            };
1343
1343            struct MouseWheelScrollView
1344            {
1344                Mouse::Wheel wheel;
1345                float delta;
1346            };
1347
1348            struct MouseWheelEvent
1349            {
1349                int delta;
1350               int x;
1351              int y;
1352            };
1353
1353            struct MouseWheelScrollView
1354            {
1354                Mouse::Wheel wheel;
1355                float delta;
1356            };
1357
1358            struct MouseWheelEvent
1359            {
1359                int delta;
1360               int x;
1361              int y;
1362            };
1363
1363            struct MouseWheelScrollView
1364            {
1364                Mouse::Wheel wheel;
1365                float delta;
1366            };
1367
1368            struct MouseWheelEvent
1369            {
1369                int delta;
1370               int x;
1371              int y;
1372            };
1373
1373            struct MouseWheelScrollView
1374            {
1374                Mouse::Wheel wheel;
1375                float delta;
1376            };
1377
1378            struct MouseWheelEvent
1379            {
1379                int delta;
1380               int x;
1381              int y;
1382            };
1383
1383            struct MouseWheelScrollView
1384            {
1384                Mouse::Wheel wheel;
1385                float delta;
1386            };
1387
1388            struct MouseWheelEvent
1389            {
1389                int delta;
1390               int x;
1391              int y;
1392            };
1393
1393            struct MouseWheelScrollView
1394            {
1394                Mouse::Wheel wheel;
1395                float delta;
1396            };
1397
1398            struct MouseWheelEvent
1399            {
1399                int delta;
1400               int x;
1401              int y;
1402            };
1403
1399            struct MouseWheelScrollView
1400            {
1400                Mouse::Wheel wheel;
1401                float delta;
1402            };
1403
1404            struct MouseWheelEvent
1405            {
1405                int delta;
1406               int x;
1407              int y;
1408            };
1409
1409            struct MouseWheelScrollView
1410            {
1410                Mouse::Wheel wheel;
1411                float delta;
1412            };
1413
1414            struct MouseWheelEvent
1415            {
1415                int delta;
1416               int x;
1417              int y;
1418            };
1419
1419            struct MouseWheelScrollView
1420            {
1420                Mouse::Wheel wheel;
1421                float delta;
1422            };
1423
1424            struct MouseWheelEvent
1425            {
1425                int delta;
1426               int x;
1427              int y;
1428            };
1429
1429            struct MouseWheelScrollView
1430            {
1430                Mouse::Wheel wheel;
1431                float delta;
1432            };
1433
1434            struct MouseWheelEvent
1435            {
1435                int delta;
1436               int x;
1437              int y;
1438            };
1439
1439            struct MouseWheelScrollView
1440            {
1440                Mouse::Wheel wheel;
1441                float delta;
1442            };
1443
1444            struct MouseWheelEvent
1445            {
1445                int delta;
1446               int x;
1447              int y;
1448            };
1449
1449            struct MouseWheelScrollView
1450            {
1450                Mouse::Wheel wheel;
1451                float delta;
1452            };
1453
1454            struct MouseWheelEvent
1455            {
1455                int delta;
1456               int x;
1457              int y;
1458            };
1459
1459            struct MouseWheelScrollView
1460            {
1460                Mouse::Wheel wheel;
1461                float delta;
1462            };
1463
1464            struct MouseWheelEvent
1465            {
1465                int delta;
1466               int x;
1467              int y;
1468            };
1469
1469            struct MouseWheelScrollView
1470            {
1470                Mouse::Wheel wheel;
1471                float delta;
1472            };
1473
1474            struct MouseWheelEvent
1475            {
1475                int delta;
1476               int x;
1477              int y;
1478            };
1479
1479            struct MouseWheelScrollView
1480            {
1480                Mouse::Wheel wheel;
1481                float delta;
1482            };
1483
1484            struct MouseWheelEvent
1485            {
1485                int delta;
1486               int x;
1487              int y;
1488            };
1489
1489            struct MouseWheelScrollView
1490            {
1490                Mouse::Wheel wheel;
1491                float delta;
1492            };
1493
1494            struct MouseWheelEvent
1495            {
1495                int delta;
1496               int x;
1497              int y;
1498            };
1499
1499            struct MouseWheelScrollView
1500            {
1500                Mouse::Wheel wheel;
1501                float delta;
1502            };
1503
1504            struct MouseWheelEvent
1505            {
1505                int delta;
1506               int x;
1507              int y;
1508            };
1509
1509            struct MouseWheelScrollView
1510            {
1510                Mouse::Wheel wheel;
1511                float delta;
1512            };
1513
1514            struct MouseWheelEvent
1515            {
1515                int delta;
1516               int x;
1517              int y;
1518            };
151
```

```
124         int x;
125         int y;
126     };
127
133     struct JoystickConnectEvent
134     {
135         unsigned int joystickId;
136     };
137
142     struct JoystickMoveEvent
143     {
144         unsigned int joystickId;
145         Joystick::Axis axis;
146         float position;
147     };
148
154     struct JoystickButtonEvent
155     {
156         unsigned int joystickId;
157         unsigned int button;
158     };
159
164     struct TouchEvent
165     {
166         unsigned int finger;
167         int x;
168         int y;
169     };
170
175     struct SensorEvent
176     {
177         Sensor::Type type;
178         float x;
179         float y;
180         float z;
181     };
182
187     enum EventType
188     {
189         Closed,
190         Resized,
191         LostFocus,
192         GainedFocus,
193         TextEntered,
194         KeyPressed,
195         KeyReleased,
196         MouseWheelMoved,
197         MouseWheelScrolled,
198         MouseButtonPressed,
199         MouseButtonReleased,
200         MouseMoved,
201         MouseEntered,
```

```

202     MouseLeft,
203     JoystickButtonPressed,
204     JoystickButtonReleased,
205     JoystickMoved,
206     JoystickConnected,
207     JoystickDisconnected,
208     TouchBegan,
209     TouchMoved,
210     TouchEnded,
211     SensorChanged,
212
213     Count
214 };
215
216 // Member data
217 EventType type;
218
219 union
220 {
221     SizeEvent size;
222     KeyEvent key;
223     TextEvent text;
224     MouseMoveEvent mouseMove;
225     MouseButtonEvent mouseButton;
226     MouseWheelEvent mouseWheel;
227     MouseWheelScrollEvent mouseWheelScroll;
228     JoystickMoveEvent joystickMove;
229     JoystickButtonEvent joystickButton;
230     JoystickConnectEvent joystickConnect;
231     TouchEvent touch;
232     SensorEvent sensor;
233 };
234 };
235 };
236 };
237
238 } // namespace sf
239
240
241 #endif // SFML_EVENT_HPP
242
243

```

Audio/Export.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_AUDIO_EXPORT_HPP
26 #define SFML_AUDIO_EXPORT_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32
33
35 // Define portable import / export macros
37 #if defined(SFML_AUDIO_EXPORTS)
```

```
38     #define SFML_AUDIO_API SFML_API_EXPORT
40
41 #else
42
43     #define SFML_AUDIO_API SFML_API_IMPORT
44
45 #endif
46
47
48 #endif // SFML_AUDIO_EXPORT_HPP
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

Graphics/Export.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_GRAPHICS_EXPORT_HPP
26 #define SFML_GRAPHICS_EXPORT_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32
33
35 // Define portable import / export macros
37 #if defined(SFML_GRAPHICS_EXPORTS)
```

```
38     #define SFML_GRAPHICS_API SFML_API_EXPORT
40
41 #else
42
43     #define SFML_GRAPHICS_API SFML_API_IMPORT
44
45 #endif
46
47
48 #endif // SFML_GRAPHICS_EXPORT_HPP
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

Network/Export.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_NETWORK_EXPORT_HPP
26 #define SFML_NETWORK_EXPORT_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32
33
35 // Define portable import / export macros
37 #if defined(SFML_NETWORK_EXPORTS)
```

```
38     #define SFML_NETWORK_API SFML_API_EXPORT
40
41 #else
42
43     #define SFML_NETWORK_API SFML_API_IMPORT
44
45 #endif
46
47
48 #endif // SFML_NETWORK_EXPORT_HPP
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

System/Export.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SYSTEM_EXPORT_HPP
26 #define SFML_SYSTEM_EXPORT_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32
33
35 // Define portable import / export macros
37 #if defined(SFML_SYSTEM_EXPORTS)
```

```
38     #define SFML_SYSTEM_API SFML_API_EXPORT
40
41 #else
42
43     #define SFML_SYSTEM_API SFML_API_IMPORT
44
45 #endif
46
47
48 #endif // SFML_SYSTEM_EXPORT_HPP
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

Window/Export.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_WINDOW_EXPORT_HPP
26 #define SFML_WINDOW_EXPORT_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32
33
35 // Define portable import / export macros
37 #if defined(SFML_WINDOW_EXPORTS)
```

```
38     #define SFML_WINDOW_API SFML_API_EXPORT
40
41 #else
42
43     #define SFML_WINDOW_API SFML_API_IMPORT
44
45 #endif
46
47
48 #endif // SFML_WINDOW_EXPORT_HPP
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

FileInputStream.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_FILEINPUTSTREAM_HPP
26 #define SFML_FILEINPUTSTREAM_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32 #include <SFML/System/Export.hpp>
33 #include <SFML/System/InputStream.hpp>
34 #include <SFML/System/NonCopyable.hpp>
35 #include <cstdio>
```

```
36 #include <string>
37
38 #ifdef ANDROID
39 namespace sf
40 {
41 namespace priv
42 {
43 class SFML_SYSTEM_API ResourceStream;
44 }
45 }
46 #endif
47
48
49 namespace sf
50 {
55 class SFML_SYSTEM_API FileInputStream : public InputStream, Non
56 {
57 public:
62     FileInputStream();
63
68     virtual ~FileInputStream();
69
78     bool open(const std::string& filename);
79
92     virtual Int64 read(void* data, Int64 size);
93
102    virtual Int64 seek(Int64 position);
103
110    virtual Int64 tell();
111
118    virtual Int64 getSize();
119
120 private:
121
123     // Member data
125 #ifdef ANDROID
126     priv::ResourceStream* m_file;
127 #else
128     std::FILE* m_file;
129 #endif
130 };
131
132 } // namespace sf
133
134
135 #endif // SFML_FILEINPUTSTREAM_HPP
136
137
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

Font.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
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 class SFML_GRAPHICS_API Font
51 {
52 public:
53
58     struct Info
59     {
60         std::string family;
61     };
62
63 public:
64
71     Font();
72
79     Font(const Font& copy);
80
87     ~Font();
88
109    bool loadFromFile(const std::string& filename);
110
130    bool loadFromMemory(const void* data, std::size_t sizeInByte
131
152    bool loadFromStream(InputStream& stream);
153
160    const Info& getInfo() const;
161
180    const Glyph& getGlyph(Uint32 codePoint, unsigned int characterSize, unsigned int outlineThickness = 0) const;
181
198    float getKerning(Uint32 first, Uint32 second, unsigned int characterSize) const;
199
211    float getLineSpacing(unsigned int characterSize) const;
212
226    float getUnderlinePosition(unsigned int characterSize) const;
227
240    float getUnderlineThickness(unsigned int characterSize) const;
241
254    const Texture& getTexture(unsigned int characterSize) const;
255
264    Font& operator =(const Font& right);
265
266 private:
267
```

```

272     struct Row
273     {
274         Row(unsigned int rowTop, unsigned int rowHeight) : width
275             height(rowHeight) {}
276         unsigned int width;
277         unsigned int top;
278         unsigned int height;
279     };
280
282     // Types
284     typedef std::map<Uint64, Glyph> GlyphTable;
285
286     struct Page
287     {
288         Page();
289
290         GlyphTable      glyphs;
291         Texture        texture;
292         unsigned int   nextRow;
293         std::vector<Row> rows;
294     };
295
296     void cleanup();
297
298     Glyph loadGlyph(Uint32 codePoint, unsigned int characterSize
299                      outlineThickness) const;
300
301     IntRect findGlyphRect(Page& page, unsigned int width, unsigned
302                           int height) const;
303
304     bool setCurrentSize(unsigned int characterSize) const;
305
306     // Types
307     typedef std::map<unsigned int, Page> PageTable;
308
309     // Member data
310     void*          m_library;
311     void*          m_face;
312     void*          m_streamRec;
313     void*          m_stroker;
314     int*           mRefCount;
315     Info           m_info;
316     mutable PageTable m_pages;
317     mutable std::vector<Uint8> m_pixelBuffer;
318 #ifdef SFML_SYSTEM_ANDROID
319     void*          m_stream;
320 #endif
321 };
322 } // namespace sf
323
324

```

```
365 #endif // SFML_FONT_HPP
366
367
```

SFML is licensed under the terms and conditions of the [zlib/png license](#).

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#).

Ftp.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_FTP_HPP
26 #define SFML_FTP_HPP
27
29 // Headers
31 #include <SFML/Network/Export.hpp>
32 #include <SFML/Network/TcpSocket.hpp>
33 #include <SFML/System/NonCopyable.hpp>
34 #include <SFML/System/Time.hpp>
35 #include <string>
```

```
36 #include <vector>
37
38
39 namespace sf
40 {
41     class IpAddress;
42
43     class SFML_NETWORK_API Ftp : NonCopyable
44     {
45         public:
46
47             enum TransferMode
48             {
49                 Binary,
50                 Ascii,
51                 Ebcdic
52             };
53
54             class SFML_NETWORK_API Response
55             {
56                 public:
57
58                 enum Status
59                 {
60                     // 1xx: the requested action is being initiated,
61                     // expect another reply before proceeding with a new
62                     RestartMarkerReply      = 110,
63                     ServiceReadySoon        = 120,
64                     DataConnectionAlreadyOpened = 125,
65                     OpeningDataConnection    = 150,
66
67                     // 2xx: the requested action has been successfully completed
68                     Ok                      = 200,
69                     PointlessCommand         = 202,
70                     SystemStatus              = 211,
71                     DirectoryStatus           = 212,
72                     FileStatus                = 213,
73                     HelpMessage               = 214,
74                     SystemType                = 215,
75                     ServiceReady               = 220,
76                     ClosingConnection          = 221,
77                     DataConnectionOpened       = 225,
78                     ClosingDataConnection      = 226,
79                     EnteringPassiveMode        = 227,
80                     LoggedIn                  = 230,
81                     FileActionOk               = 250,
82                     DirectoryOk                = 257,
83
84                     // 3xx: the command has been accepted, but the request
85                     // is dormant, pending receipt of further information
86                     NeedPassword               = 331,
87                     NeedAccountToLogIn         = 332,
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
```

```

104     NeedInformation      = 350,
105
106     // 4xx: the command was not accepted and the request
107     // but the error condition is temporary and the acti
108     ServiceUnavailable    = 421,
109     DataConnectionUnavailable = 425,
110     TransferAborted       = 426,
111     FileActionAborted     = 450,
112     LocalError             = 451,
113     InsufficientStorageSpace = 452,
114
115     // 5xx: the command was not accepted and
116     // the requested action did not take place
117     CommandUnknown         = 500,
118     ParametersUnknown      = 501,
119     CommandNotImplemented   = 502,
120     BadCommandSequence     = 503,
121     ParameterNotImplemented = 504,
122     NotLoggedIn            = 530,
123     NeedAccountToStore     = 532,
124     FileUnavailable        = 550,
125     PageTypeUnknown        = 551,
126     NotEnoughMemory        = 552,
127     FilenameNotAllowed     = 553,
128
129     // 10xx: SFML custom codes
130     InvalidResponse        = 1000,
131     ConnectionFailed       = 1001,
132     ConnectionClosed        = 1002,
133     InvalidFile             = 1003
134 };
135
136     explicit Response(Status code = InvalidResponse, const std::string& message) :
137     m_status{code}, m_message{message}
138 {
139     if (code == InvalidResponse)
140         isOk() = false;
141     else
142         isOk() = true;
143 }
144
145     bool isOk() const;
146
147     Status getStatus() const;
148
149     const std::string& getMessage() const;
150
151     private:
152
153     // Member data
154     Status      m_status;
155     std::string  m_message;
156 };
157
158     class SFML_NETWORK_API DirectoryResponse : public Response
159     {
160     public:
161
162         DirectoryResponse(const Response& response);
163
164     };

```

```
199
206     const std::string& getDirectory() const;
207
208 private:
209
211 // Member data
213     std::string m_directory;
214 };
215
216
221 class SFML_NETWORK_API ListingResponse : public Response
222 {
223 public:
224
232     ListingResponse(const Response& response, const std::sti
233
240     const std::vector<std::string>& getList() const;
241
242 private:
243
245 // Member data
247     std::vector<std::string> m_listing;
248 };
249
250
258 ~Ftp();
259
281     Response connect(const IpAddress& server, unsigned short port,
Time::Zero);
282
291     Response disconnect();
292
302     Response login();
303
316     Response login(const std::string& name, const std::string& p
317
327     Response keepAlive();
328
340     DirectoryResponse getWorkingDirectory();
341
357     ListingResponse getDirectoryListing(const std::string& direc
358
371     Response changeDirectory(const std::string& directory);
372
381     Response parentDirectory();
382
396     Response createDirectory(const std::string& name);
397
413     Response deleteDirectory(const std::string& name);
414
429     Response renameFile(const std::string& file, const std::strin
430
```

```
446     Response deleteFile(const std::string& name);
447
468     Response download(const std::string& remoteFile, const std::string mode = Binary);
469
487     Response upload(const std::string& localFile, const std::string mode = Binary);
488
505     Response sendCommand(const std::string& command, const std::string mode = Binary);
506
507 private:
508
518     Response getResponse();
519
525     class DataChannel;
526
527     friend class DataChannel;
528
530     // Member data
532     TcpSocket m_commandSocket;
533     std::string m_receiveBuffer;
534 };
535
536 } // namespace sf
537
538
539 #endif // SFML_FTP_HPP
540
541
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

GIResource.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_GLRESOURCE_HPP
26 #define SFML_GLRESOURCE_HPP
27
29 // Headers
31 #include <SFML/Window/Export.hpp>
32 #include <SFML/System/NonCopyable.hpp>
33
34
35 namespace sf
```

```
36  {
37
38 class Context;
39
40 class SFML_WINDOW_API GlResource
41 {
42 protected:
43
44     GlResource();
45
46     ~GlResource();
47
48     static void ensureGlContext();
49
50     class SFML_WINDOW_API TransientContextLock : NonCopyable
51     {
52         public:
53             TransientContextLock();
54
55             ~TransientContextLock();
56
57         private:
58             Context* m_context;
59         };
60     };
61
62 } // namespace sf
63
64
65 #endif // SFML_GLRESOURCE_HPP
66
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Gsl.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_GLSL_HPP
26 #define SFML_GLSL_HPP
27
29 // Headers
31 #include <SFML/Graphics/Transform.hpp>
32 #include <SFML/Graphics/Color.hpp>
33 #include <SFML/System/Vector2.hpp>
34 #include <SFML/System/Vector3.hpp>
35
```

```
36
37 namespace sf
38 {
39 namespace priv
40 {
41     // Forward declarations
42     template <std::size_t Columns, std::size_t Rows>
43     struct Matrix;
44
45     template <typename T>
46     struct Vector4;
47
48 #include <SFML/Graphics/Glsl.inl>
49
50 } // namespace priv
51
52
57 namespace Glsl
58 {
59
64     typedef Vector2<float> Vec2;
65
70     typedef Vector2<int> Ivec2;
71
76     typedef Vector2<bool> Bvec2;
77
82     typedef Vector3<float> Vec3;
83
88     typedef Vector3<int> Ivec3;
89
94     typedef Vector3<bool> Bvec3;
95
96 #ifdef SFML_DOXYGEN
97
110     typedef implementation-defined Vec4;
111
124     typedef implementation-defined Ivec4;
125
130     typedef implementation-defined Bvec4;
131
155     typedef implementation-defined Mat3;
156
181     typedef implementation-defined Mat4;
182
183 #else // SFML_DOXYGEN
184
185     typedef priv::Vector4<float> Vec4;
186     typedef priv::Vector4<int> Ivec4;
187     typedef priv::Vector4<bool> Bvec4;
188     typedef priv::Matrix<3, 3> Mat3;
189     typedef priv::Matrix<4, 4> Mat4;
190
```

```
191 #endif // SFML_DOXYGEN
192
193 } // namespace Gsl
194 } // namespace sf
195
196 #endif // SFML_GLSL_HPP
197
198
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Glyph.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_GLYPH_HPP
26 #define SFML_GLYPH_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Rect.hpp>
33
34
35 namespace sf
```

```
36 {
41 class SFML_GRAPHICS_API Glyph
42 {
43 public:
44
49     Glyph() : advance(0) {}
50
52     // Member data
54     float advance;
55     FloatRect bounds;
56     IntRect textureRect;
57 };
58
59 } // namespace sf
60
61
62 #endif // SFML_GLYPH_HPP
63
64
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Graphics.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_GRAPHICS_HPP
26 #define SFML_GRAPHICS_HPP
27
29 // Headers
31
32 #include <SFML/Window.hpp>
33 #include <SFML/Graphics/BlendMode.hpp>
34 #include <SFML/Graphics/CircleShape.hpp>
35 #include <SFML/Graphics/Color.hpp>
```

```
36 #include <SFML/Graphics/ConvexShape.hpp>
37 #include <SFML/Graphics/Drawable.hpp>
38 #include <SFML/Graphics/Font.hpp>
39 #include <SFML/Graphics/Glyph.hpp>
40 #include <SFML/Graphics/Image.hpp>
41 #include <SFML/Graphics/PrimitiveType.hpp>
42 #include <SFML/Graphics/Rect.hpp>
43 #include <SFML/Graphics/RectangleShape.hpp>
44 #include <SFML/Graphics/RenderStates.hpp>
45 #include <SFML/Graphics/RenderTarget.hpp>
46 #include <SFML/Graphics/RenderTexture.hpp>
47 #include <SFML/Graphics/RenderWindow.hpp>
48 #include <SFML/Graphics/Shader.hpp>
49 #include <SFML/Graphics/Shape.hpp>
50 #include <SFML/Graphics/Sprite.hpp>
51 #include <SFML/Graphics/Text.hpp>
52 #include <SFML/Graphics/Texture.hpp>
53 #include <SFML/Graphics/Transform.hpp>
54 #include <SFML/Graphics/Transformable.hpp>
55 #include <SFML/Graphics/Vertex.hpp>
56 #include <SFML/Graphics/VertexArray.hpp>
57 #include <SFML/Graphics/View.hpp>
58
59
60 #endif // SFML_GRAPHICS_HPP
61
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Http.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_HTTP_HPP
26 #define SFML_HTTP_HPP
27
29 // Headers
31 #include <SFML/Network/Export.hpp>
32 #include <SFML/Network/IpAddress.hpp>
33 #include <SFML/Network/TcpSocket.hpp>
34 #include <SFML/System/NonCopyable.hpp>
35 #include <SFML/System/Time.hpp>
```

```
36 #include <map>
37 #include <string>
38
39
40 namespace sf
41 {
46 class SFML_NETWORK_API Http : NonCopyable
47 {
48 public:
49
54     class SFML_NETWORK_API Request
55     {
56     public:
57
62         enum Method
63         {
64             Get,
65             Post,
66             Head,
67             Put,
68             Delete
69         };
70
82     Request(const std::string& uri = "/", Method method = Ge
    "");
83
97     void setField(const std::string& field, const std::strin
98
109     void setMethod(Method method);
110
121     void setUri(const std::string& uri);
122
132     void setHttpVersion(unsigned int major, unsigned int min
133
144     void setBody(const std::string& body);
145
146 private:
147
148     friend class Http;
149
159     std::string prepare() const;
160
171     bool hasField(const std::string& field) const;
172
174     // Types
176     typedef std::map<std::string, std::string> FieldTable;
177
179     // Member data
181     FieldTable m_fields;
182     Method m_method;
183     std::string m_uri;
184     unsigned int m_majorVersion;
```

```
185         unsigned int m_minorVersion;
186         std::string m_body;
187     };
188
189     class SFML_NETWORK_API Response
190     {
191     public:
192
193     enum Status
194     {
195         // 2xx: success
196         Ok          = 200,
197         Created      = 201,
198         Accepted     = 202,
199         NoContent    = 204,
200         ResetContent = 205,
201         PartialContent = 206,
202
203         // 3xx: redirection
204         MultipleChoices = 300,
205         MovedPermanently = 301,
206         MovedTemporarily = 302,
207         NotModified     = 304,
208
209         // 4xx: client error
210         BadRequest     = 400,
211         Unauthorized   = 401,
212         Forbidden      = 403,
213         NotFound       = 404,
214         RangeNotSatisfiable = 407,
215
216         // 5xx: server error
217         InternalServerError = 500,
218         NotImplemented    = 501,
219         BadGateway        = 502,
220         ServiceUnavailable = 503,
221         GatewayTimeout    = 504,
222         VersionNotSupported = 505,
223
224         // 10xx: SFML custom codes
225         InvalidResponse   = 1000,
226         ConnectionFailed  = 1001
227     };
228
229     Response();
230
231     const std::string& getField(const std::string& field) const;
232
233     Status getStatus() const;
234
235     unsigned int getMajorHttpVersion() const;
236
237     void setField(const std::string& field, const std::string& value);
238
239     void setStatus(Status status);
240
241     void setBody(const std::string& body);
242
243     std::string getBody() const;
244
245     void minorVersion(unsigned int minorVersion);
246
247     unsigned int getMinorVersion() const;
248
249     void majorHttpVersion(unsigned int majorVersion);
250
251     unsigned int getMajorHttpVersion() const;
252
253     void setConnectionFailed();
254
255     void clear();
256
257     void setInternalServerError();
258
259     void setNotImplemented();
260
261     void setBadGateway();
262
263     void setServiceUnavailable();
264
265     void setGatewayTimeout();
266
267     void setVersionNotSupported();
268
269     void setMultipleChoices();
270
271     void setMovedPermanently();
272
273     void setMovedTemporarily();
274
275     void setNotModified();
276
277     void setBadRequest();
278
279     void setUnauthorized();
280
281     void setForbidden();
282
283     void setNotFound();
284
285     void setRangeNotSatisfiable();
```

```

290         unsigned int getMinorHttpVersion() const;
291
294     const std::string& getBody() const;
295
296     private:
297
298         friend class Http;
299
300     void parse(const std::string& data);
301
302
303     void parseFields(std::istream &in);
304
305
306     // Types
307     typedef std::map<std::string, std::string> FieldTable;
308
309
310     // Member data
311     FieldTable m_fields;
312     Status m_status;
313     unsigned int m_majorVersion;
314     unsigned int m_minorVersion;
315     std::string m_body;
316 };
317
318
319     Http();
320
321
322     Http(const std::string& host, unsigned short port = 0);
323
324     void setHost(const std::string& host, unsigned short port =
325
326
327     Response sendRequest(const Request& request, Time timeout =
328
329
330     private:
331
332
333     // Member data
334     TcpSocket m_connection;
335     IpAddress m_host;
336     std::string mHostName;
337     unsigned short m_port;
338 };
339
340
341 } // namespace sf
342
343
344 #endif // SFML_HTTP_HPP
345
346
347

```

Copyright © Laurent Gomila :: Documentation generated by doxygen

Image.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_IMAGE_HPP
26 #define SFML_IMAGE_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Color.hpp>
33 #include <SFML/Graphics/Rect.hpp>
34 #include <string>
35 #include <vector>
```

```
36
37
38 namespace sf
39 {
40 class InputStream;
41
46 class SFML_GRAPHICS_API Image
47 {
48 public:
49
56     Image();
57
62     ~Image();
63
72     void create(unsigned int width, unsigned int height, const Color& color);
73
87     void create(unsigned int width, unsigned int height, const Uint8* pixels);
88
104    bool loadFromFile(const std::string& filename);
105
122    bool loadFromMemory(const void* data, std::size_t size);
123
139    bool loadFromStream(InputStream& stream);
140
156    bool saveToFile(const std::string& filename) const;
157
164    Vector2u getSize() const;
165
177    void createMaskFromColor(const Color& color, Uint8 alpha = 0xFF);
178
199    void copy(const Image& source, unsigned int destX, unsigned int destY,
200              const IntRect& sourceRect = IntRect(0, 0, 0, 0), bool applyAlpha = false);
201
215    void setPixel(unsigned int x, unsigned int y, const Color& color);
216
232    Color getPixel(unsigned int x, unsigned int y) const;
233
247    const Uint8* getPixelsPtr() const;
248
253    void flipHorizontally();
254
259    void flipVertically();
260
261 private:
262
264     // Member data
266     Vector2u          m_size;
267     std::vector<Uint8> m_pixels;
268     #ifdef SFML_SYSTEM_ANDROID
269     void*             m_stream;
270     #endif
271 };
```

```
272
273 } // namespace sf
274
275
276 #endif // SFML_IMAGE_HPP
277
278
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

InputSoundFile.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_INPUTSOUNDFILE_HPP
26 #define SFML_INPUTSOUNDFILE_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <SFML/System/NonCopyable.hpp>
33 #include <SFML/System/Time.hpp>
34 #include <string>
35
```

```
36
37 namespace sf
38 {
39 class InputStream;
40 class SoundFileReader;
41
46 class SFML_AUDIO_API InputSoundFile : NonCopyable
47 {
48 public:
49
54     InputSoundFile();
55
60     ~InputSoundFile();
61
73     bool openFromFile(const std::string& filename);
74
87     bool openFromMemory(const void* data, std::size_t sizeInByte
88
100    bool openFromStream(InputStream& stream);
101
112    bool openForWriting(const std::string& filename, unsigned in
sampleRate);
113
120    Uint64 getSampleCount() const;
121
128    unsigned int getChannelCount() const;
129
136    unsigned int getSampleRate() const;
137
147    Time getDuration() const;
148
165    void seek(Uint64 sampleOffset);
166
179    void seek(Time timeOffset);
180
190    Uint64 read(Int16* samples, Uint64 maxCount);
191
192 private:
193
198     void close();
199
201     // Member data
203     SoundFileReader* m_reader;
204     InputStream*      m_stream;
205     bool              m_streamOwned;
206     Uint64            m_sampleCount;
207     unsigned int      m_channelCount;
208     unsigned int      m_sampleRate;
209 };
210
211 } // namespace sf
212
```

```
213  
214 #endif // SFML_INPUTSOUNDFILE_HPP  
215  
216
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

InputStream.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_INPUTSTREAM_HPP
26 #define SFML_INPUTSTREAM_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32 #include <SFML/System/Export.hpp>
33
34
35 namespace sf
```

```
36 {
41 class SFML_SYSTEM_API InputStream
42 {
43 public:
44
49     virtual ~InputStream() {}
50
63     virtual Int64 read(void* data, Int64 size) = 0;
64
73     virtual Int64 seek(Int64 position) = 0;
74
81     virtual Int64 tell() = 0;
82
89     virtual Int64 getSize() = 0;
90 };
91
92 } // namespace sf
93
94
95 #endif // SFML_INPUTSTREAM_HPP
96
97
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

IpAddress.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_IPADDRESS_HPP
26 #define SFML_IPADDRESS_HPP
27
29 // Headers
31 #include <SFML/Network/Export.hpp>
32 #include <SFML/System/Time.hpp>
33 #include <iostream>
34 #include <ostream>
35 #include <string>
```

```
36
37
38 namespace sf
39 {
40     class SFML_NETWORK_API IpAddress
41     {
42         public:
43
44             IpAddress();
45
46             IpAddress(const std::string& address);
47
48             IpAddress(const char* address);
49
50             IpAddress(Uint8 byte0, Uint8 byte1, Uint8 byte2, Uint8 byte3);
51
52             explicit IpAddress(Uint32 address);
53
54             std::string toString() const;
55
56             Uint32 toInteger() const;
57
58             static IpAddress getLocalAddress();
59
60             static IpAddress getPublicAddress(Time timeout = Time::Zero);
61
62             // Static member data
63             static const IpAddress None;
64             static const IpAddress Any;
65             static const IpAddress LocalHost;
66             static const IpAddress Broadcast;
67
68     private:
69
70         friend SFML_NETWORK_API bool operator <(const IpAddress& left,
71                                         const IpAddress& right);
72
73         void resolve(const std::string& address);
74
75         // Member data
76         Uint32 m_address;
77         bool m_valid;
78     };
79
80     SFML_NETWORK_API bool operator ==(const IpAddress& left, const IpAddress& right);
81
82     SFML_NETWORK_API bool operator !=(const IpAddress& left, const IpAddress& right);
83
84     SFML_NETWORK_API bool operator <(const IpAddress& left, const IpAddress& right);
85
86     SFML_NETWORK_API bool operator >(const IpAddress& left, const IpAddress& right);
87
88     SFML_NETWORK_API bool operator <=(const IpAddress& left, const IpAddress& right);
89
90     SFML_NETWORK_API bool operator >=(const IpAddress& left, const IpAddress& right);
91 }
```

```
262
272 SFML_NETWORK_API bool operator >=(const IpAddress& left, const ]
273
283 SFML_NETWORK_API std::istream& operator >>(std::istream& stream,
284
294 SFML_NETWORK_API std::ostream& operator <<(std::ostream& stream,
295
296 } // namespace sf
297
298
299 #endif // SFML_IPADDRESS_HPP
300
301
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Joystick.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_JOYSTICK_HPP
26 #define SFML_JOYSTICK_HPP
27
29 // Headers
31 #include <SFML/Window/Export.hpp>
32 #include <SFML/System/String.hpp>
33
34
35 namespace sf
```

```
36  {
41  class SFML_WINDOW_API Joystick
42  {
43  public:
44
49      enum
50      {
51          Count      = 8,
52          ButtonCount = 32,
53          AxisCount   = 8
54      };
55
60      enum Axis
61      {
62          X,
63          Y,
64          Z,
65          R,
66          U,
67          V,
68          PovX,
69          PovY
70      };
71
76      struct SFML_WINDOW_API Identification
77      {
78          Identification();
79
80          String name;
81          unsigned int vendorId;
82          unsigned int productId;
83      };
84
93      static bool isConnected(unsigned int joystick);
94
105     static unsigned int getButtonCount(unsigned int joystick);
106
118     static bool hasAxis(unsigned int joystick, Axis axis);
119
131     static bool isButtonPressed(unsigned int joystick, unsigned
132
144     static float getAxisPosition(unsigned int joystick, Axis axis);
145
154     static Identification getIdentification(unsigned int joystick);
155
165     static void update();
166 };
167
168 } // namespace sf
169
170
171 #endif // SFML_JOYSTICK_HPP
```

172

173

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Keyboard.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_KEYBOARD_HPP
26 #define SFML_KEYBOARD_HPP
27
29 // Headers
31 #include <SFML/Window/Export.hpp>
32
33
34 namespace sf
35 {
```

```
40 class SFML_WINDOW_API Keyboard
41 {
42 public:
43
48     enum Key
49     {
50         Unknown = -1,
51         A = 0,
52         B,
53         C,
54         D,
55         E,
56         F,
57         G,
58         H,
59         I,
60         J,
61         K,
62         L,
63         M,
64         N,
65         O,
66         P,
67         Q,
68         R,
69         S,
70         T,
71         U,
72         V,
73         W,
74         X,
75         Y,
76         Z,
77         Num0,
78         Num1,
79         Num2,
80         Num3,
81         Num4,
82         Num5,
83         Num6,
84         Num7,
85         Num8,
86         Num9,
87         Escape,
88         LControl,
89         LShift,
90         LAlt,
91         LSystem,
92         RControl,
93         RShift,
94         RAlt,
95         RSystem,
```

96	Menu,
97	LBracket,
98	RBracket,
99	SemiColon,
100	Comma,
101	Period,
102	Quote,
103	Slash,
104	BackSlash,
105	Tilde,
106	Equal,
107	Dash,
108	Space,
109	Return,
110	BackSpace,
111	Tab,
112	PageUp,
113	PageDown,
114	End,
115	Home,
116	Insert,
117	Delete,
118	Add,
119	Subtract,
120	Multiply,
121	Divide,
122	Left,
123	Right,
124	Up,
125	Down,
126	Numpad0,
127	Numpad1,
128	Numpad2,
129	Numpad3,
130	Numpad4,
131	Numpad5,
132	Numpad6,
133	Numpad7,
134	Numpad8,
135	Numpad9,
136	F1,
137	F2,
138	F3,
139	F4,
140	F5,
141	F6,
142	F7,
143	F8,
144	F9,
145	F10,
146	F11,
147	F12,

```
148     F13,
149     F14,
150     F15,
151     Pause,
152
153     KeyCount
154 };
155
164     static bool isKeyPressed(Key key);
165
179     static void setVirtualKeyboardVisible(bool visible);
180 };
181
182 } // namespace sf
183
184
185 #endif // SFML_KEYBOARD_HPP
186
187
```

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Listener.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_LISTENER_HPP
26 #define SFML_LISTENER_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <SFML/System/Vector3.hpp>
33
34
35 namespace sf
```

```
36  {
42  class SFML_AUDIO_API Listener
43  {
44  public:
45
58      static void setGlobalVolume(float volume);
59
68      static float getGlobalVolume();
69
82      static void setPosition(float x, float y, float z);
83
94      static void setPosition(const Vector3f& position);
95
104     static Vector3f getPosition();
105
123     static void setDirection(float x, float y, float z);
124
140     static void setDirection(const Vector3f& direction);
141
150     static Vector3f getDirection();
151
169     static void setUpVector(float x, float y, float z);
170
186     static void setUpVector(const Vector3f& upVector);
187
196     static Vector3f getUpVector();
197 };
198
199 } // namespace sf
200
201
202 #endif // SFML_LISTENER_HPP
203
204
```

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

Lock.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_LOCK_HPP
26 #define SFML_LOCK_HPP
27
29 // Headers
31 #include <SFML/System/Export.hpp>
32 #include <SFML/System/NonCopyable.hpp>
33
34
35 namespace sf
```

```
36 {
37 class Mutex;
38
43 class SFML_SYSTEM_API Lock : NonCopyable
44 {
45 public:
46
55     explicit Lock(Mutex& mutex);
56
63     ~Lock();
64
65 private:
66
68     // Member data
70     Mutex& m_mutex;
71 };
72
73 } // namespace sf
74
75
76 #endif // SFML_LOCK_HPP
77
78
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Main.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_MAIN_HPP
26 #define SFML_MAIN_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32
33
34 #if defined(SFML_SYSTEM_IOS)
35
```

```
36     // On iOS, we have no choice but to have our own main,  
37     // so we need to rename the user one and call it later  
38     #define main sfmlMain  
39  
40 #endif  
41  
42  
43 #endif // SFML_MAIN_HPP
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

Files

[File List](#)

mainpage.hpp

1

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

MemoryInputStream.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_MEMORYINPUTSTREAM_HPP
26 #define SFML_MEMORYINPUTSTREAM_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32 #include <SFML/System/InputStream.hpp>
33 #include <SFML/System/Export.hpp>
34 #include <cstdlib>
35
```

```
36
37 namespace sf
38 {
39 class SFML_SYSTEM_API MemoryInputStream : public InputStream
40 {
41 public:
42
43     MemoryInputStream();
44
45     void open(const void* data, std::size_t sizeInBytes);
46
47     virtual Int64 read(void* data, Int64 size);
48
49     virtual Int64 seek(Int64 position);
50
51     virtual Int64 tell();
52
53     virtual Int64 getSize();
54
55 private:
56
57     // Member data
58     const char* m_data;
59     Int64      m_size;
60     Int64      m_offset;
61 };
62
63 } // namespace sf
64
65 #endif // SFML_MEMORYINPUTSTREAM_HPP
66
67
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Mouse.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_MOUSE_HPP
26 #define SFML_MOUSE_HPP
27
29 // Headers
31 #include <SFML/Window/Export.hpp>
32 #include <SFML/System/Vector2.hpp>
33
34
35 namespace sf
```

```
36  {
37  class Window;
38
43  class SFML_WINDOW_API Mouse
44  {
45 public:
46
51     enum Button
52     {
53         Left,
54         Right,
55         Middle,
56         XButton1,
57         XButton2,
58
59         ButtonCount
60     };
61
66     enum Wheel
67     {
68         VerticalWheel,
69         HorizontalWheel
70     };
71
80     static bool isButtonPressed(Button button);
81
91     static Vector2i getPosition();
92
104    static Vector2i getPosition(const Window& relativeTo);
105
115    static void setPosition(const Vector2i& position);
116
127    static void setPosition(const Vector2i& position, const Window&
128 );
129
130 } // namespace sf
131
132
133 #endif // SFML_MOUSE_HPP
134
135
```

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

Music.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_MUSIC_HPP
26 #define SFML_MUSIC_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <SFML/Audio/SoundStream.hpp>
33 #include <SFML/Audio/InputSoundFile.hpp>
34 #include <SFML/System/Mutex.hpp>
35 #include <SFML/System/Time.hpp>
```

```
36 #include <string>
37 #include <vector>
38
39
40 namespace sf
41 {
42 class InputStream;
43
44 class SFML_AUDIO_API Music : public SoundStream
45 {
46 public:
47
48     Music();
49
50     ~Music();
51
52     bool openFromFile(const std::string& filename);
53
54     bool openFromMemory(const void* data, std::size_t sizeInByte
55 );
56
57     bool openFromStream(InputStream& stream);
58
59     Time getDuration() const;
60
61 protected:
62
63     virtual bool onGetData(Chunk& data);
64
65     virtual void onSeek(Time timeOffset);
66
67 private:
68
69     void initialize();
70
71     // Member data
72     InputSoundFile m_file;
73     Time m_duration;
74     std::vector<Int16> m_samples;
75     Mutex m_mutex;
76 };
77 } // namespace sf
78
79
80 #endif // SFML_MUSIC_HPP
81
82
```

Copyright © Laurent Gomila :: Documentation generated by d

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Mutex.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_MUTEX_HPP
26 #define SFML_MUTEX_HPP
27
29 // Headers
31 #include <SFML/System/Export.hpp>
32 #include <SFML/System/NonCopyable.hpp>
33
34
35 namespace sf
```

```
36  {
37  namespace priv
38  {
39      class MutexImpl;
40  }
41
47  class SFML_SYSTEM_API Mutex : NonCopyable
48  {
49 public:
50
55     Mutex();
56
61     ~Mutex();
62
73     void lock();
74
81     void unlock();
82
83 private:
84
86     // Member data
88     priv::MutexImpl* m_mutexImpl;
89 }
90
91 } // namespace sf
92
93
94 #endif // SFML_MUTEX_HPP
95
96
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

NativeActivity.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_NATIVEACTIVITY_HPP
26 #define SFML_NATIVEACTIVITY_HPP
27
28
30 // Headers
32 #include <SFML/System/Export.hpp>
33
34
35 #if !defined(SFML_SYSTEM_ANDROID)
```

```
36 #error NativeActivity.hpp: This header is Android only.
37 #endif
38
39
40 struct ANativeActivity;
41
42 namespace sf
43 {
44     SFML_SYSTEM_API ANativeActivity* getNativeActivity();
45 }
46 // namespace sf
47
48
49 #endif // SFML_NATIVEACTIVITY_HPP
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Network.hpp

```
1 //  
3 // SFML - Simple and Fast Multimedia Library  
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)  
5 //  
6 // This software is provided 'as-is', without any express or imp  
7 // In no event will the authors be held liable for any damages a  
software.  
8 //  
9 // Permission is granted to anyone to use this software for any  
10 // including commercial applications, and to alter it and redist  
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 n  
17 //  
18 // 2. Altered source versions must be plainly marked as such,  
19 //     and must not be misrepresented as being the original softw  
20 //  
21 // 3. This notice may not be removed or altered from any source  
22 //  
24  
25 #ifndef SFML_NETWORK_HPP  
26 #define SFML_NETWORK_HPP  
27  
29 // Headers  
31  
32 #include <SFML/System.hpp>  
33 #include <SFML/Network/Ftp.hpp>  
34 #include <SFML/Network/Http.hpp>  
35 #include <SFML/Network/IpAddress.hpp>
```

```
36 #include <SFML/Network/Packet.hpp>
37 #include <SFML/Network/Socket.hpp>
38 #include <SFML/Network/SocketHandle.hpp>
39 #include <SFML/Network/SocketSelector.hpp>
40 #include <SFML/Network/TcpListener.hpp>
41 #include <SFML/Network/TcpSocket.hpp>
42 #include <SFML/Network/UdpSocket.hpp>
43
44
45 #endif // SFML_NETWORK_HPP
46
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

NonCopyable.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_NOCOPYABLE_HPP
26 #define SFML_NOCOPYABLE_HPP
27
29 // Headers
31 #include <SFML/System/Export.hpp>
32
33
34 namespace sf
35 {
```

```
41 class SFML_SYSTEM_API NonCopyable
42 {
43 protected:
44
45     NonCopyable() {}
54
55 private:
56
57     NonCopyable(const NonCopyable&);
58
59     NonCopyable& operator =(const NonCopyable&);
60 };
61
62 } // namespace sf
63
64
65 #endif // SFML_NONCOPYABLE_HPP
66
67
68
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

OpenGL.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_OPENGL_HPP
26 #define SFML_OPENGL_HPP
27
28
32 #include <SFML/Config.hpp>
33
34
39 #if defined(SFML_SYSTEM_WINDOWS)
40
```

```
41     // The Visual C++ version of gl.h uses WINGDIAPI and APIENTR
42 #ifdef _MSC_VER
43     #include <windows.h>
44 #endif
45
46     #include <GL/gl.h>
47
48 #elif defined(SFML_SYSTEM_LINUX) || defined(SFML_SYSTEM_FREEBSD)
49
50     #if defined(SFML_OPENGL_ES)
51         #include <GLES/gl.h>
52         #include <GLES/glext.h>
53     #else
54         #include <GL/gl.h>
55     #endif
56
57 #elif defined(SFML_SYSTEM_MACOS)
58
59     #include <OpenGL/gl.h>
60
61 #elif defined(SFML_SYSTEM_IOS)
62
63     #include <OpenGLES/ES1/gl.h>
64     #include <OpenGLES/ES1/glext.h>
65
66 #elif defined(SFML_SYSTEM_ANDROID)
67
68     #include <GLES/gl.h>
69     #include <GLES/glext.h>
70
71     // We're not using OpenGL ES 2+ yet, but we can use the sRGB
72     #include <GLES2/gl2ext.h>
73
74 #endif
75
76
77 #endif // SFML_OPENGL_HPP
```

OutputSoundFile.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_OUTPUTSOUNDFILE_HPP
26 #define SFML_OUTPUTSOUNDFILE_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <SFML/System/NonCopyable.hpp>
33 #include <string>
34
35
```

```
36 namespace sf
37 {
38 class SoundFileWriter;
39
44 class SFML_AUDIO_API OutputSoundFile : NonCopyable
45 {
46 public:
47
52     OutputSoundFile();
53
60     ~OutputSoundFile();
61
74     bool openFromFile(const std::string& filename, unsigned int
channelCount);
75
83     void write(const Int16* samples, Uint64 count);
84
85 private:
86
91     void close();
92
94     // Member data
96     SoundFileWriter* m_writer;
97 };
98
99 } // namespace sf
100
101
102 #endif // SFML_OUTPUTSOUNDFILE_HPP
103
104
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Packet.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_PACKET_HPP
26 #define SFML_PACKET_HPP
27
29 // Headers
31 #include <SFML/Network/Export.hpp>
32 #include <string>
33 #include <vector>
34
35
```

```
36 namespace sf
37 {
38 class String;
39 class TcpSocket;
40 class UdpSocket;
41
47 class SFML_NETWORK_API Packet
48 {
49     // A bool-like type that cannot be converted to integer or p
50     typedef bool (Packet::*BoolType)(std::size_t);
51
52 public:
53
54     Packet();
55
56     virtual ~Packet();
57
58     void append(const void* data, std::size_t sizeInBytes);
59
60     void clear();
61
62     const void* getData() const;
63
64     std::size_t getDataSize() const;
65
66     bool endOfPacket() const;
67
68 public:
69
70     operator BoolType() const;
71
72
73     Packet& operator >>(bool& data);
74     Packet& operator >>(Int8& data);
75     Packet& operator >>(Uint8& data);
76     Packet& operator >>(Int16& data);
77     Packet& operator >>(Uint16& data);
78     Packet& operator >>(Int32& data);
79     Packet& operator >>(Uint32& data);
80     Packet& operator >>(Int64& data);
81     Packet& operator >>(Uint64& data);
82     Packet& operator >>(float& data);
83     Packet& operator >>(double& data);
84     Packet& operator >>(char* data);
85     Packet& operator >>(std::string& data);
86     Packet& operator >>(wchar_t* data);
87     Packet& operator >>(std::wstring& data);
88     Packet& operator >>(String& data);
89
90
91     Packet& operator <<(bool data);
92     Packet& operator <<(Int8 data);
93     Packet& operator <<(Uint8 data);
94     Packet& operator <<(Int16 data);
```

```

202     Packet& operator <<(Uint16)           data);
203     Packet& operator <<(Int32)            data);
204     Packet& operator <<(UInt32)           data);
205     Packet& operator <<(Int64)            data);
206     Packet& operator <<(UInt64)           data);
207     Packet& operator <<(float)             data);
208     Packet& operator <<(double)            data);
209     Packet& operator <<(const char*)       data);
210     Packet& operator <<(const std::string&) data);
211     Packet& operator <<(const wchar_t*)      data);
212     Packet& operator <<(const std::wstring&) data);
213     Packet& operator <<(const String&)        data);
214
215 protected:
216
217     friend class TcpSocket;
218     friend class UdpSocket;
219
220     virtual const void* onSend(std::size_t& size);
221
222     virtual void onReceive(const void* data, std::size_t size);
223
224 private:
225
226     bool operator ==(const Packet& right) const;
227     bool operator !=(const Packet& right) const;
228
229     bool checkSize(std::size_t size);
230
231     // Member data
232     std::vector<char> m_data;
233     std::size_t         m_readPos;
234     std::size_t         m_sendPos;
235     bool               m_isValid;
236
237 };
238
239 } // namespace sf
240
241
242 #endif // SFML_PACKET_HPP
243
244

```

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

PrimitiveType.hpp

```
1 //  
3 // SFML - Simple and Fast Multimedia Library  
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)  
5 //  
6 // This software is provided 'as-is', without any express or imp  
7 // In no event will the authors be held liable for any damages a  
software.  
8 //  
9 // Permission is granted to anyone to use this software for any  
10 // including commercial applications, and to alter it and redist  
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 n  
17 //  
18 // 2. Altered source versions must be plainly marked as such,  
19 //     and must not be misrepresented as being the original softw  
20 //  
21 // 3. This notice may not be removed or altered from any source  
22 //  
24  
25 #ifndef SFML_PRIMITIVETYPE_HPP  
26 #define SFML_PRIMITIVETYPE_HPP  
27  
28 namespace sf  
29 {  
39 enum PrimitiveType  
40 {  
41     Points,  
42     Lines,
```

```
43     LineStrip,
44     Triangles,
45     TriangleStrip,
46     TriangleFan,
47     Quads,
48
49     // Deprecated names
50     LinesStrip      = LineStrip,
51     TrianglesStrip = TriangleStrip,
52     TrianglesFan   = TriangleFan
53 };
54
55 } // namespace sf
56
57
58 #endif // SFML_PRIMITIVETYPE_HPP
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Rect.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_RECT_HPP
26 #define SFML_RECT_HPP
27
29 // Headers
31 #include <SFML/System/Vector2.hpp>
32 #include <algorithm>
33
34
35 namespace sf
```

```
36  {
41  template <typename T>
42  class Rect
43  {
44  public:
45
46      Rect();
47
48      Rect(T rectLeft, T rectTop, T rectWidth, T rectHeight);
49
50      Rect(const Vector2<T>& position, const Vector2<T>& size);
51
52      template <typename U>
53      explicit Rect(const Rect<U>& rectangle);
54
55      bool contains(T x, T y) const;
56
57      bool contains(const Vector2<T>& point) const;
58
59      bool intersects(const Rect<T>& rectangle) const;
60
61      bool intersects(const Rect<T>& rectangle, Rect<T>& intersec
62
63      // Member data
64      T left;
65      T top;
66      T width;
67      T height;
68
69  };
70
71
72  template <typename T>
73  bool operator==(const Rect<T>& left, const Rect<T>& right);
74
75
76  template <typename T>
77  bool operator!=(const Rect<T>& left, const Rect<T>& right);
78
79
80  #include <SFML/Graphics/Rect.inl>
81
82
83  // Create typedefs for the most common types
84  typedef Rect<int> IntRect;
85  typedef Rect<float> FloatRect;
86
87
88  } // namespace sf
89
90
91
92  #endif // SFML_RECT_HPP
93
94
95
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

RectangleShape.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_RECTANGLESHAPE_HPP
26 #define SFML_RECTANGLESHAPE_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Shape.hpp>
33
34
35 namespace sf
```

```
36  {
41  class SFML_GRAPHICS_API RectangleShape : public Shape
42  {
43  public:
44
51      explicit RectangleShape(const Vector2f& size = Vector2f(0,
52
61      void setSize(const Vector2f& size);
62
71      const Vector2f& getSize() const;
72
80      virtual std::size_t getPointCount() const;
81
95      virtual Vector2f getPoint(std::size_t index) const;
96
97 private:
98
100     // Member data
102     Vector2f m_size;
103 };
104
105 } // namespace sf
106
107
108 #endif // SFML_RECTANGLESHAPE_HPP
109
110
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

RenderStates.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_RENDERSTATES_HPP
26 #define SFML_RENDERSTATES_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/BlendMode.hpp>
33 #include <SFML/Graphics/Transform.hpp>
34
35
```

```
36 namespace sf
37 {
38 class Shader;
39 class Texture;
40
45 class SFML_GRAPHICS_API RenderStates
46 {
47 public:
48
61     RenderStates();
62
69     RenderStates(const BlendMode& theBlendMode);
70
77     RenderStates(const Transform& theTransform);
78
85     RenderStates(const Texture* theTexture);
86
93     RenderStates(const Shader* theShader);
94
104    RenderStates(const BlendMode& theBlendMode, const Transform&
105                  const Texture* theTexture, const Shader* theSha
106
108    // Static member data
110    static const RenderStates Default;
111
113    // Member data
115    BlendMode blendMode;
116    Transform transform;
117    const Texture* texture;
118    const Shader* shader;
119 }
120
121 } // namespace sf
122
123
124 #endif // SFML_RENDERSTATES_HPP
125
126
```

RenderTarget.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_RENDERTARGET_HPP
26 #define SFML_RENDERTARGET_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Color.hpp>
33 #include <SFML/Graphics/Rect.hpp>
34 #include <SFML/Graphics/View.hpp>
35 #include <SFML/Graphics/Transform.hpp>
```

```
36 #include <SFML/Graphics/BlendMode.hpp>
37 #include <SFML/Graphics/RenderStates.hpp>
38 #include <SFML/Graphics/PrimitiveType.hpp>
39 #include <SFML/Graphics/Vertex.hpp>
40 #include <SFML/System/NonCopyable.hpp>
41
42
43 namespace sf
44 {
45 class Drawable;
46
51 class SFML_GRAPHICS_API RenderTarget : NonCopyable
52 {
53 public:
54
59     virtual ~RenderTarget();
60
70     void clear(const Color& color = Color(0, 0, 0, 255));
71
72     void setView(const View& view);
73
101    const View& getView() const;
102
114    const View& getDefaultView() const;
115
129    IntRect getViewport(const View& view) const;
130
149    Vector2f mapPixelToCoords(const Vector2i& point) const;
150
180    Vector2f mapPixelToCoords(const Vector2i& point, const View& view) const;
181
200    Vector2i mapCoordsToPixel(const Vector2f& point) const;
201
227    Vector2i mapCoordsToPixel(const Vector2f& point, const View& view) const;
228
236    void draw(const Drawable& drawable, const RenderStates& states = RenderStates());
237
247    void draw(const Vertex* vertices, std::size_t vertexCount,
248              PrimitiveType type, const RenderStates& states = RenderStates());
249
256    virtual Vector2u getSize() const = 0;
257
290    void pushGLStates();
291
301    void popGLStates();
302
324    void resetGLStates();
325
326 protected:
327
332     RenderTarget();
333
```

```

341     void initialize();
342
343 private:
344
349     void applyCurrentView();
350
357     void applyBlendMode(const BlendMode& mode);
358
365     void applyTransform(const Transform& transform);
366
373     void applyTexture(const Texture* texture);
374
381     void applyShader(const Shader* shader);
382
395     virtual bool activate(bool active) = 0;
396
401     struct StatesCache
402     {
403         enum {VertexCacheSize = 4};
404
405         bool glStatesSet;
406         bool viewChanged;
407         BlendMode lastBlendMode;
408         Uint64 lastTextureId;
409         bool useVertexCache;
410         Vertex vertexCache[VertexCacheSize];
411     };
412
414     // Member data
416     View m_defaultView;
417     View m_view;
418     StatesCache m_cache;
419 };
420
421 } // namespace sf
422
423
424 #endif // SFML_RENDERTARGET_HPP
425
426

```

RenderTexture.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_RENDERTEXTURE_HPP
26 #define SFML_RENDERTEXTURE_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Texture.hpp>
33 #include <SFML/Graphics/RenderTarget.hpp>
34
35
```

```
36 namespace sf
37 {
38 namespace priv
39 {
40     class RenderTextureImpl;
41 }
42
47 class SFML_GRAPHICS_API RenderTexture : public RenderTarget
48 {
49 public:
50
60     RenderTexture();
61
66     virtual ~RenderTexture();
67
86     bool create(unsigned int width, unsigned int height, bool de
87
99     void setSmooth(bool smooth);
100
109    bool isSmooth() const;
110
122    void setRepeated(bool repeated);
123
132    bool isRepeated() const;
133
148    bool generateMipmap();
149
165    bool setActive(bool active = true);
166
176    void display();
177
187    virtual Vector2u getSize() const;
188
203    const Texture& getTexture() const;
204
205 private:
206
218    virtual bool activate(bool active);
219
221    // Member data
223    priv::RenderTextureImpl* m_impl;
224    Texture                  m_texture;
225 };
226
227 } // namespace sf
228
229
230 #endif // SFML_RENDERTEXTURE_HPP
231
232
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

RenderWindow.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_RENDERWINDOW_HPP
26 #define SFML_RENDERWINDOW_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/RenderTarget.hpp>
33 #include <SFML/Graphics/Image.hpp>
34 #include <SFML/Window/Window.hpp>
35 #include <string>
```

```
36
37
38 namespace sf
39 {
40     class SFML_GRAPHICS_API RenderWindow : public Window, public RenderWindow
41     {
42         public:
43             RenderWindow();
44
45             RenderWindow(VideoMode mode, const String& title, Uint32 style,
46             ContextSettings& settings = ContextSettings());
47
48             explicit RenderWindow(WindowHandle handle, const ContextSettings&
49             ContextSettings());
50
51             virtual ~RenderWindow();
52
53             virtual Vector2u getSize() const;
54
55             SFML_DEPRECATED Image capture() const;
56
57         protected:
58
59             virtual void onCreate();
60
61             virtual void onResize();
62
63     private:
64
65             virtual bool activate(bool active);
66     };
67
68 } // namespace sf
69
70 #endif // SFML_RENDERWINDOW_HPP
71
72
73
```

Sensor.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SENSOR_HPP
26 #define SFML_SENSOR_HPP
27
29 // Headers
31 #include <SFML/Window/Export.hpp>
32 #include <SFML/System/Vector3.hpp>
33 #include <SFML/System/Time.hpp>
34
35
```

```
36 namespace sf
37 {
42 class SFML_WINDOW_API Sensor
43 {
44 public:
45
50     enum Type
51     {
52         Accelerometer,
53         Gyroscope,
54         Magnetometer,
55         Gravity,
56         UserAcceleration,
57         Orientation,
58
59         Count
60     };
61
70     static bool isAvailable(Type sensor);
71
85     static void setEnabled(Type sensor, bool enabled);
86
95     static Vector3f getValue(Type sensor);
96 };
97
98 } // namespace sf
99
100
101 #endif // SFML_SENSOR_HPP
102
103
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

Shader.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SHADER_HPP
26 #define SFML_SHADER_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Glsl.hpp>
33 #include <SFML/Window/GlResource.hpp>
34 #include <SFML/System/NonCopyable.hpp>
35 #include <SFML/System/Vector2.hpp>
```

```
36 #include <SFML/System/Vector3.hpp>
37 #include <map>
38 #include <string>
39
40
41 namespace sf
42 {
43     class Color;
44     class InputStream;
45     class Texture;
46     class Transform;
47
48     class SFML_GRAPHICS_API Shader : GlResource, NonCopyable
49     {
50     public:
51
52         enum Type
53         {
54             Vertex,
55             Geometry,
56             Fragment
57         };
58
59         struct CurrentTextureType {};
60
61         static CurrentTextureType CurrentTexture;
62
63     public:
64
65         Shader();
66
67         ~Shader();
68
69         bool loadFromFile(const std::string& filename, Type type);
70
71         bool loadFromFile(const std::string& vertexShaderFilename, c
72                         fragmentShaderFilename);
73
74         bool loadFromFile(const std::string& vertexShaderFilename, c
75                         geometryShaderFilename, const std::string& fragmentShaderFilen
76
77         bool loadFromMemory(const std::string& shader, Type type);
78
79         bool loadFromMemory(const std::string& vertexShader, const s
80
81         bool loadFromMemory(const std::string& vertexShader, const s
82                         const std::string& fragmentShader);
83
84         bool loadFromStream(InputStream& stream, Type type);
85
86         bool loadFromStream(InputStream& vertexShaderStream, InputSt
```

```
288     bool loadFromStream(InputStream& vertexShaderStream, InputStream& fragmentShaderStream);
289
297     void setUniform(const std::string& name, float x);
298
306     void setUniform(const std::string& name, const Glsl::Vec2& v);
307
315     void setUniform(const std::string& name, const Glsl::Vec3& v);
316
333     void setUniform(const std::string& name, const Glsl::Vec4& v);
334
342     void setUniform(const std::string& name, int x);
343
351     void setUniform(const std::string& name, const Glsl::Ivec2& v);
352
360     void setUniform(const std::string& name, const Glsl::Ivec3& v);
361
377     void setUniform(const std::string& name, const Glsl::Ivec4& v);
378
386     void setUniform(const std::string& name, bool x);
387
395     void setUniform(const std::string& name, const Glsl::Bvec2& v);
396
404     void setUniform(const std::string& name, const Glsl::Bvec3& v);
405
413     void setUniform(const std::string& name, const Glsl::Bvec4& v);
414
422     void setUniform(const std::string& name, const Glsl::Mat3& m);
423
431     void setUniform(const std::string& name, const Glsl::Mat4& m);
432
463     void setUniform(const std::string& name, const Texture& texture);
464
486     void setUniform(const std::string& name, CurrentTextureType);
487
496     void setUniformArray(const std::string& name, const float* s, length);
497
506     void setUniformArray(const std::string& name, const Glsl::Vec2& v, length);
507
516     void setUniformArray(const std::string& name, const Glsl::Vec3& v, length);
517
526     void setUniformArray(const std::string& name, const Glsl::Vec4& v, length);
527
536     void setUniformArray(const std::string& name, const Glsl::Mat3& m, length);
537
546     void setUniformArray(const std::string& name, const Glsl::Mat4& m, length);
```

```
547     SFML_DEPRECATED void setParameter(const std::string& name, f
554     SFML_DEPRECATED void setParameter(const std::string& name, f
555     SFML_DEPRECATED void setParameter(const std::string& name, f
562     SFML_DEPRECATED void setParameter(const std::string& name, f
563     SFML_DEPRECATED void setParameter(const std::string& name, f
570     SFML_DEPRECATED void setParameter(const std::string& name, f
571     SFML_DEPRECATED void setParameter(const std::string& name, f
578     SFML_DEPRECATED void setParameter(const std::string& name, f
      w);
579     SFML_DEPRECATED void setParameter(const std::string& name, c
586     SFML_DEPRECATED void setParameter(const std::string& name, c
587     SFML_DEPRECATED void setParameter(const std::string& name, c
594     SFML_DEPRECATED void setParameter(const std::string& name, c
595     SFML_DEPRECATED void setParameter(const std::string& name, c
602     SFML_DEPRECATED void setParameter(const std::string& name, c
603     SFML_DEPRECATED void setParameter(const std::string& name, c
610     SFML_DEPRECATED void setParameter(const std::string& name, c
611     SFML_DEPRECATED void setParameter(const std::string& name, c
618     SFML_DEPRECATED void setParameter(const std::string& name, c
619     SFML_DEPRECATED void setParameter(const std::string& name, c
626     SFML_DEPRECATED void setParameter(const std::string& name, c
627     SFML_DEPRECATED void setParameter(const std::string& name, c
638     unsigned int getNativeHandle() const;
639
661     static void bind(const Shader* shader);
662
673     static bool isAvailable();
674
692     static bool isGeometryAvailable();
693
694 private:
695
709     bool compile(const char* vertexShaderCode, const char* geomet
      fragmentShaderCode);
710
718     void bindTextures() const;
719
728     int getUniformLocation(const std::string& name);
729
737     struct UniformBinder;
738
740     // Types
742     typedef std::map<int, const Texture*> TextureTable;
743     typedef std::map<std::string, int> UniformTable;
744
746     // Member data
748     unsigned int m_shaderProgram;
749     int          m_currentTexture;
750     TextureTable m_textures;
751     UniformTable m_uniforms;
752 };
753
```

```
754 } // namespace sf
755
756
757 #endif // SFML_SHADER_HPP
758
759
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

Shape.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SHAPE_HPP
26 #define SFML_SHAPE_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Drawable.hpp>
33 #include <SFML/Graphics/Transformable.hpp>
34 #include <SFML/Graphics/VertexArray.hpp>
35 #include <SFML/System/Vector2.hpp>
```

```
36
37
38 namespace sf
39 {
40     class SFML_GRAPHICS_API Shape : public Drawable, public Transformable
41     {
42         public:
43
44             virtual ~Shape();
45
46             void setTexture(const Texture* texture, bool resetRect = false);
47
48             void setTextureRect(const IntRect& rect);
49
50             void setFillColor(const Color& color);
51
52             void setOutlineColor(const Color& color);
53
54             void setOutlineThickness(float thickness);
55
56             const Texture* getTexture() const;
57
58             const IntRect& getTextureRect() const;
59
60             const Color& getFillColor() const;
61
62             const Color& getOutlineColor() const;
63
64             float getOutlineThickness() const;
65
66             virtual std::size_t getPointCount() const = 0;
67
68             virtual Vector2f getPoint(std::size_t index) const = 0;
69
70             FloatRect getLocalBounds() const;
71
72             FloatRect getGlobalBounds() const;
73
74         protected:
75
76             Shape();
77
78             void update();
79
80         private:
81
82             virtual void draw(RenderTarget& target, RenderStates states) const;
83
84             void updateFillColors();
85
86             void updateTexCoords();
87
88         };
89
90     };
91 }
```

```
295     void updateOutline();
296
297     void updateOutlineColors();
298
299 private:
300
301     // Member data
302     const Texture* m_texture;
303     IntRect          m_textureRect;
304     Color            m_fillColor;
305     Color            m_outlineColor;
306     float             m_outlineThickness;
307     VertexArray      m_vertices;
308     VertexArray      m_outlineVertices;
309     FloatRect        m_insideBounds;
310     FloatRect        m_bounds;
311 };
312
313 } // namespace sf
314
315 #endif // SFML_SHAPE_HPP
316
317
318
319
320
321
322
323
324
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Sleep.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SLEEP_HPP
26 #define SFML_SLEEP_HPP
27
29 // Headers
31 #include <SFML/System/Export.hpp>
32 #include <SFML/System/Time.hpp>
33
34
35 namespace sf
```

```
36 {
37 void SFML_SYSTEM_API sleep(Time duration);
38
39 } // namespace sf
40
41
42 #endif // SFML_SLEEP_HPP
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

Socket.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SOCKET_HPP
26 #define SFML_SOCKET_HPP
27
29 // Headers
31 #include <SFML/Network/Export.hpp>
32 #include <SFML/Network/SocketHandle.hpp>
33 #include <SFML/System/NonCopyable.hpp>
34 #include <vector>
35
```

```
36
37 namespace sf
38 {
39 class SocketSelector;
40
45 class SFML_NETWORK_API Socket : NonCopyable
46 {
47 public:
48
53     enum Status
54     {
55         Done,
56         NotReady,
57         Partial,
58         Disconnected,
59         Error
60     };
61
66     enum
67     {
68         AnyPort = 0
69     };
70
71 public:
72
77     virtual ~Socket();
78
96     void setBlocking(bool blocking);
97
106    bool isBlocking() const;
107
108 protected:
109
114     enum Type
115     {
116         Tcp,
117         Udp
118     };
119
128     Socket(Type type);
129
140     SocketHandle getHandle() const;
141
148     void create();
149
159     void create(SocketHandle handle);
160
167     void close();
168
169 private:
170
171     friend class SocketSelector;
```

```
172
174     // Member data
176     Type          m_type;
177     SocketHandle  m_socket;
178     bool          m_isBlocking;
179 };
180
181 } // namespace sf
182
183
184 #endif // SFML_SOCKET_HPP
185
186
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SocketHandle.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SOCKETHANDLE_HPP
26 #define SFML_SOCKETHANDLE_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32
33 #if defined(SFML_SYSTEM_WINDOWS)
34     #include <basetsd.h>
35 #endif
```

```
36
37
38 namespace sf
39 {
40     // Define the low-level socket handle type, specific to
41     // each platform
42 #if defined(SFML_SYSTEM_WINDOWS)
43     typedef UINT_PTR SocketHandle;
44 #else
45     typedef int SocketHandle;
46 #endif
47 }
48 } // namespace sf
49
50
51
52 #endif
53
54 } // namespace sf
55
56
57 #endif // SFML_SOCKETHANDLE_HPP
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

SocketSelector.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SOCKETSELECTOR_HPP
26 #define SFML_SOCKETSELECTOR_HPP
27
29 // Headers
31 #include <SFML/Network/Export.hpp>
32 #include <SFML/System/Time.hpp>
33
34
35 namespace sf
```

```
36  {
37  class Socket;
38
43  class SFML_NETWORK_API SocketSelector
44  {
45  public:
46
51      SocketSelector();
52
59      SocketSelector(const SocketSelector& copy);
60
65      ~SocketSelector();
66
80      void add(Socket& socket);
81
93      void remove(Socket& socket);
94
105     void clear();
106
123     bool wait(Time timeout = Time::Zero);
124
142     bool isReady(Socket& socket) const;
143
152     SocketSelector& operator =(const SocketSelector& right);
153
154 private:
155
156     struct SocketSelectorImpl;
157
159     // Member data
161     SocketSelectorImpl* m_impl;
162 };
163
164 } // namespace sf
165
166
167 #endif // SFML_SOCKETSELECTOR_HPP
168
169
```

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

Sound.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SOUND_HPP
26 #define SFML_SOUND_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <SFML/Audio/SoundSource.hpp>
33 #include <SFML/System/Time.hpp>
34 #include <cstdlib>
35
```

```
36
37 namespace sf
38 {
39 class SoundBuffer;
40
45 class SFML_AUDIO_API Sound : public SoundSource
46 {
47 public:
48
53     Sound();
54
61     explicit Sound(const SoundBuffer& buffer);
62
69     Sound(const Sound& copy);
70
75     ~Sound();
76
89     void play();
90
100    void pause();
101
112    void stop();
113
126    void setBuffer(const SoundBuffer& buffer);
127
141    void setLoop(bool loop);
142
156    void setPlayingOffset(Time timeOffset);
157
164    const SoundBuffer* getBuffer() const;
165
174    bool getLoop() const;
175
184    Time getPlayingOffset() const;
185
192    Status getStatus() const;
193
202    Sound& operator =(const Sound& right);
203
213    void resetBuffer();
214
215 private:
216
218     // Member data
220     const SoundBuffer* m_buffer;
221 };
222
223 } // namespace sf
224
225
226 #endif // SFML_SOUND_HPP
227
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SoundBuffer.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SOUNDBUFFER_HPP
26 #define SFML_SOUNDBUFFER_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <SFML/Audio/AIResource.hpp>
33 #include <SFML/System/Time.hpp>
34 #include <string>
35 #include <vector>
```

```
36 #include <set>
37
38
39 namespace sf
40 {
41     class Sound;
42     class InputSoundFile;
43     class InputStream;
44
45     class SFML_AUDIO_API SoundBuffer : AlResource
46     {
47     public:
48
49         SoundBuffer();
50
51         SoundBuffer(const SoundBuffer& copy);
52
53         ~SoundBuffer();
54
55         bool loadFromFile(const std::string& filename);
56
57         bool loadFromMemory(const void* data, std::size_t sizeInByte);
58
59         bool loadFromStream(InputStream& stream);
60
61         bool loadFromSamples(const Int16* samples, Uint64 sampleCount,
62                             unsigned int sampleRate);
63
64         bool saveToFile(const std::string& filename) const;
65
66         const Int16* getSamples() const;
67
68         Uint64 getSampleCount() const;
69
70         unsigned int getSampleRate() const;
71
72         unsigned int getChannelCount() const;
73
74         Time getDuration() const;
75
76         SoundBuffer& operator =(const SoundBuffer& right);
77
78     private:
79
80         friend class Sound;
81
82         bool initialize(InputSoundFile& file);
83
84         bool update(unsigned int channelCount, unsigned int sampleRate);
85
86         void attachSound(Sound* sound) const;
87
88 }
```

```
265     void detachSound( Sound* sound) const;
266
267     // Types
268     typedef std::set<Sound*> SoundList;
269
270     // Member data
271     unsigned int          m_buffer;
272     std::vector<Int16>    m_samples;
273     Time                 m_duration;
274     mutable SoundList   m_sounds;
275 };
276
277 } // namespace sf
278
279
280 #endif // SFML_SOUNDBUFFER_HPP
281
282
283
284
285
286
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

SoundBufferRecorder.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SOUNDBUFFERRECORDER_HPP
26 #define SFML_SOUNDBUFFERRECORDER_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <SFML/Audio/SoundBuffer.hpp>
33 #include <SFML/Audio/SoundRecorder.hpp>
34 #include <vector>
35
```

```
36
37 namespace sf
38 {
39     class SFML_AUDIO_API SoundBufferRecorder : public SoundRecorder
40     {
41     public:
42         ~SoundBufferRecorder();
43
44         const SoundBuffer& getBuffer() const;
45
46     protected:
47
48         virtual bool onStart();
49
50         virtual bool onProcessSamples(const Int16* samples, std::size_t
51                                     count);
52
53         virtual void onStop();
54
55     private:
56
57         // Member data
58         std::vector<Int16> m_samples;
59         SoundBuffer          m_buffer;
60     };
61
62
63 } // namespace sf
64
65 #endif // SFML_SOUNDBUFFERRECORDER_HPP
66
67
68
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SoundFileFactory.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SOUNDFILEFACTORY_HPP
26 #define SFML_SOUNDFILEFACTORY_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <string>
33 #include <vector>
34
35
```

```
36 namespace sf
37 {
38 class InputStream;
39 class SoundFileReader;
40 class SoundFileWriter;
41
46 class SFML_AUDIO_API SoundFileFactory
47 {
48 public:
49
56     template <typename T>
57     static void registerReader();
58
65     template <typename T>
66     static void unregisterReader();
67
74     template <typename T>
75     static void registerWriter();
76
83     template <typename T>
84     static void unregisterWriter();
85
98     static SoundFileReader* createReaderFromFilename(const std::
99
113     static SoundFileReader* createReaderFromMemory(const void* c
114
127     static SoundFileReader* createReaderFromStream(InputStream&
128
139     static SoundFileWriter* createWriterFromFilename(const std::
140
141 private:
142
144     // Types
146     struct ReaderFactory
147     {
148         bool (*check)(InputStream&);
149         SoundFileReader* (*create)();
150     };
151     typedef std::vector<ReaderFactory> ReaderFactoryArray;
152
153     struct WriterFactory
154     {
155         bool (*check)(const std::string&);
156         SoundFileWriter* (*create)();
157     };
158     typedef std::vector<WriterFactory> WriterFactoryArray;
159
161     // Static member data
163     static ReaderFactoryArray s_readers;
164     static WriterFactoryArray s_writers;
165 };
166
```

```
167 } // namespace sf
168
169 #include <SFML/Audio/SoundFileFactory.inl>
170
171 #endif // SFML_SOUNDFILEFACTORY_HPP
172
173
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

SoundFileReader.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SOUNDFILEREADER_HPP
26 #define SFML_SOUNDFILEREADER_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <string>
33
34
35 namespace sf
```

```
36  {
37  class InputStream;
38
43  class SFML_AUDIO_API SoundFileReader
44  {
45  public:
46
51      struct Info
52      {
53          Uint64      sampleCount;
54          unsigned int channelCount;
55          unsigned int sampleRate;
56      };
57
62      virtual ~SoundFileReader() {}
63
77      virtual bool open(InputStream& stream, Info& info) = 0;
78
91      virtual void seek(Uint64 sampleOffset) = 0;
92
102     virtual Uint64 read(Int16* samples, Uint64 maxCount) = 0;
103 };
104
105 } // namespace sf
106
107
108 #endif // SFML_SOUNDFILEREADER_HPP
109
110
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

SoundFileWriter.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SOUNDFILEWRITER_HPP
26 #define SFML_SOUNDFILEWRITER_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <string>
33
34
35 namespace sf
```

```
36 {
41 class SFML_AUDIO_API SoundFileWriter
42 {
43 public:
44
49     virtual ~SoundFileWriter() {}
50
61     virtual bool open(const std::string& filename, unsigned int
channelCount) = 0;
62
70     virtual void write(const Int16* samples, Uint64 count) = 0;
71 };
72
73 } // namespace sf
74
75
76 #endif // SFML_SOUNDFILEWRITER_HPP
77
78
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SoundRecorder.hpp

```
1 //  
3 // SFML - Simple and Fast Multimedia Library  
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)  
5 //  
6 // This software is provided 'as-is', without any express or imp  
7 // In no event will the authors be held liable for any damages a  
software.  
8 //  
9 // Permission is granted to anyone to use this software for any  
10 // including commercial applications, and to alter it and redist  
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 n  
17 //  
18 // 2. Altered source versions must be plainly marked as such,  
19 //     and must not be misrepresented as being the original softw  
20 //  
21 // 3. This notice may not be removed or altered from any source  
22 //  
24  
25 #ifndef SFML_SOUNDRECORDER_HPP  
26 #define SFML_SOUNDRECORDER_HPP  
27  
29 // Headers  
31 #include <SFML/Audio/Export.hpp>  
32 #include <SFML/Audio/AIResource.hpp>  
33 #include <SFML/System/Thread.hpp>  
34 #include <SFML/System/Time.hpp>  
35 #include <vector>
```

```
36 #include <string>
37
38
39 namespace sf
40 {
45 class SFML_AUDIO_API SoundRecorder : AlResource
46 {
47 public:
48
53     virtual ~SoundRecorder();
54
77     bool start(unsigned int sampleRate = 44100);
78
85     void stop();
86
97     unsigned int getSampleRate() const;
98
108    static std::vector<std::string> getAvailableDevices();
109
120    static std::string getDefaultDevice();
121
137    bool setDevice(const std::string& name);
138
145    const std::string& getDevice() const;
146
160    void setChannelCount(unsigned int channelCount);
161
173    unsigned int getChannelCount() const;
174
186    static bool isAvailable();
187
188 protected:
189
196     SoundRecorder();
197
214     void setProcessingInterval(Time interval);
215
227     virtual bool onStart();
228
243     virtual bool onProcessSamples(const Int16* samples, std::size_t
244
254     virtual void onStop();
255
256 private:
257
265     void record();
266
275     void processCapturedSamples();
276
283     void cleanup();
284
286 // Member data
```

```
288     Thread           m_thread;
289     std::vector<Int16> m_samples;
290     unsigned int       m_sampleRate;
291     Time              m_processingInterval;
292     bool               m_isCapturing;
293     std::string        m_deviceName;
294     unsigned int       m_channelCount;
295 };
296
297 } // namespace sf
298
299
300 #endif // SFML_SOUNDRECORDER_HPP
301
302
```

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

SoundSource.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SOUND SOURCE_HPP
26 #define SFML_SOUND SOURCE_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <SFML/Audio/AIResource.hpp>
33 #include <SFML/System/Vector3.hpp>
34
35
```

```
36 namespace sf
37 {
42 class SFML_AUDIO_API SoundSource : AlResource
43 {
44 public:
45
50     enum Status
51     {
52         Stopped,
53         Paused,
54         Playing
55     };
56
63     SoundSource(const SoundSource& copy);
64
69     virtual ~SoundSource();
70
85     void setPitch(float pitch);
86
98     void setVolume(float volume);
99
114    void setPosition(float x, float y, float z);
115
128    void setPosition(const Vector3f& position);
129
144    void setRelativeToListener(bool relative);
145
161    void setMinDistance(float distance);
162
180    void setAttenuation(float attenuation);
181
190    float getPitch() const;
191
200    float getVolume() const;
201
210    Vector3f getPosition() const;
211
221    bool isRelativeToListener() const;
222
231    float getMinDistance() const;
232
241    float getAttenuation() const;
242
251    SoundSource& operator =(const SoundSource& right);
252
253 protected:
254
261     SoundSource();
262
269     Status getStatus() const;
270
272 // Member data
```

```
274     unsigned int m_source;
275 };
276
277 } // namespace sf
278
279
280 #endif // SFML_SOUND SOURCE_HPP
281
282
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SoundStream.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SOUNDSTREAM_HPP
26 #define SFML_SOUNDSTREAM_HPP
27
29 // Headers
31 #include <SFML/Audio/Export.hpp>
32 #include <SFML/Audio/SoundSource.hpp>
33 #include <SFML/System/Thread.hpp>
34 #include <SFML/System/Time.hpp>
35 #include <SFML/System/Mutex.hpp>
```

```
36 #include <cstdlib>
37
38
39 namespace sf
40 {
45 class SFML_AUDIO_API SoundStream : public SoundSource
46 {
47 public:
48
53     struct Chunk
54     {
55         const Int16* samples;
56         std::size_t sampleCount;
57     };
58
63     virtual ~SoundStream();
64
77     void play();
78
88     void pause();
89
100    void stop();
101
110    unsigned int getChannelCount() const;
111
121    unsigned int getSampleRate() const;
122
129    Status getStatus() const;
130
144    void setPlayingOffset(Time timeOffset);
145
154    Time getPlayingOffset() const;
155
169    void setLoop(bool loop);
170
179    bool getLoop() const;
180
181 protected:
182
189     SoundStream();
190
205     void initialize(unsigned int channelCount, unsigned int samp
206
224     virtual bool onGetData(Chunk& data) = 0;
225
235     virtual void onSeek(Time timeOffset) = 0;
236
237 private:
238
246     void streamData();
247
261     bool fillAndPushBuffer(unsigned int bufferNum);
```

```
262     bool fillQueue();
272     void clearQueue();
281
282     enum
283     {
284         BufferCount = 3
285     };
286
288     // Member data
290     Thread      m_thread;
291     mutable Mutex m_threadMutex;
292     Status       m_threadStartState;
293     bool         m_isStreaming;
294     unsigned int m_buffers[BufferCount];
295     unsigned int m_channelCount;
296     unsigned int m_sampleRate;
297     Uint32       m_format;
298     bool         m_loop;
299     Uint64       m_samplesProcessed;
300     bool         m_endBuffers[BufferCount];
301 };
302
303 } // namespace sf
304
305
306 #endif // SFML_SOUNDSTREAM_HPP
307
308
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Sprite.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_SPRITE_HPP
26 #define SFML_SPRITE_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Drawable.hpp>
33 #include <SFML/Graphics/Transformable.hpp>
34 #include <SFML/Graphics/Vertex.hpp>
35 #include <SFML/Graphics/Rect.hpp>
```

```
36
37
38 namespace sf
39 {
40 class Texture;
41
47 class SFML_GRAPHICS_API Sprite : public Drawable, public Transf
48 {
49 public:
50
57     Sprite();
58
67     explicit Sprite(const Texture& texture);
68
78     Sprite(const Texture& texture, const IntRect& rectangle);
79
99     void setTexture(const Texture& texture, bool resetRect = fa]
100
113     void setTextureRect(const IntRect& rectangle);
114
128     void setColor(const Color& color);
129
142     const Texture* getTexture() const;
143
152     const IntRect& getTextureRect() const;
153
162     const Color& getColor() const;
163
176     FloatRect getLocalBounds() const;
177
190     FloatRect getGlobalBounds() const;
191
192 private:
193
201     virtual void draw(RenderTarget& target, RenderStates states)
202
207     void updatePositions();
208
213     void updateTexCoords();
214
216     // Member data
218     Vertex          m_vertices[4];
219     const Texture*   m_texture;
220     IntRect         m_textureRect;
221 };
222
223 } // namespace sf
224
225
226 #endif // SFML_SPRITE_HPP
227
228
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

String.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_STRING_HPP
26 #define SFML_STRING_HPP
27
29 // Headers
31 #include <SFML/System/Export.hpp>
32 #include <SFML/System/Utf.hpp>
33 #include <iterator>
34 #include <locale>
35 #include <string>
```



```
266     std::wstring toWideString() const;
267
276     std::basic_string<Uint8> toUtf8() const;
277
286     std::basic_string<Uint16> toUtf16() const;
287
299     std::basic_string<Uint32> toUtf32() const;
300
309     String& operator =(const String& right);
310
319     String& operator +=(const String& right);
320
332     Uint32 operator [](std::size_t index) const;
333
345     Uint32& operator [](std::size_t index);
346
355     void clear();
356
365     std::size_t getSize() const;
366
375     bool isEmpty() const;
376
387     void erase(std::size_t position, std::size_t count = 1);
388
399     void insert(std::size_t position, const String& str);
400
413     std::size_t find(const String& str, std::size_t start = 0) const;
414
427     void replace(std::size_t position, std::size_t length, const String& str);
428
439     void replace(const String& searchFor, const String& replaceWith);
440
456     String substring(std::size_t position, std::size_t length = 1) const;
457
469     const Uint32* getData() const;
470
479     Iterator begin();
480
489     ConstIterator begin() const;
490
503     Iterator end();
504
517     ConstIterator end() const;
518
519 private:
520
521     friend SFML_SYSTEM_API bool operator ==(const String& left, const String& right);
522     friend SFML_SYSTEM_API bool operator <(const String& left, const String& right);
523
525     // Member data
527     std::basic_string<Uint32> m_string;
528 };
```

```
529
540 SFML_SYSTEM_API bool operator ==(const String& left, const String& right)
541
552 SFML_SYSTEM_API bool operator !=(const String& left, const String& right)
553
564 SFML_SYSTEM_API bool operator <(const String& left, const String& right)
565
576 SFML_SYSTEM_API bool operator >(const String& left, const String& right)
577
588 SFML_SYSTEM_API bool operator <=(const String& left, const String& right)
589
600 SFML_SYSTEM_API bool operator >=(const String& left, const String& right)
601
612 SFML_SYSTEM_API String operator +(const String& left, const String& right)
613
614 #include <SFML/System/String.inl>
615
616 } // namespace sf
617
618
619 #endif // SFML_STRING_HPP
620
621
```

SFML is licensed under the terms and conditions of the zlib/png license.
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

System.hpp

```
1 //  
3 // SFML - Simple and Fast Multimedia Library  
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)  
5 //  
6 // This software is provided 'as-is', without any express or imp  
7 // In no event will the authors be held liable for any damages a  
software.  
8 //  
9 // Permission is granted to anyone to use this software for any  
10 // including commercial applications, and to alter it and redist  
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 n  
17 //  
18 // 2. Altered source versions must be plainly marked as such,  
19 //     and must not be misrepresented as being the original softw  
20 //  
21 // 3. This notice may not be removed or altered from any source  
22 //  
24  
25 #ifndef SFML_SYSTEM_HPP  
26 #define SFML_SYSTEM_HPP  
27  
29 // Headers  
31  
32 #include <SFML/Config.hpp>  
33 #include <SFML/System/Clock.hpp>  
34 #include <SFML/System/Err.hpp>  
35 #include <SFML/System/FileInputStream.hpp>
```

```
36 #include <SFML/System/InputStream.hpp>
37 #include <SFML/System/Lock.hpp>
38 #include <SFML/System/MemoryInputStream.hpp>
39 #include <SFML/System/Mutex.hpp>
40 #include <SFML/System/NonCopyable.hpp>
41 #include <SFML/System/Sleep.hpp>
42 #include <SFML/System/String.hpp>
43 #include <SFML/System/Thread.hpp>
44 #include <SFML/System/ThreadLocal.hpp>
45 #include <SFML/System/ThreadLocalPtr.hpp>
46 #include <SFML/System/Time.hpp>
47 #include <SFML/System/Utf.hpp>
48 #include <SFML/System/Vector2.hpp>
49 #include <SFML/System/Vector3.hpp>
50
51 #endif // SFML_SYSTEM_HPP
52
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

TcpListener.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_TCPLISTENER_HPP
26 #define SFML_TCPLISTENER_HPP
27
29 // Headers
31 #include <SFML/Network/Export.hpp>
32 #include <SFML/Network/Socket.hpp>
33 #include <SFML/Network/IpAddress.hpp>
34
35
```

```
36 namespace sf
37 {
38     class TcpSocket;
39
44     class SFML_NETWORK_API TcpListener : public Socket
45     {
46         public:
47
52         TcpListener();
53
65         unsigned short getLocalPort() const;
66
83         Status listen(unsigned short port, const IpAddress& address
84
94         void close();
95
109        Status accept(TcpSocket& socket);
110    };
111
112
113 } // namespace sf
114
115
116 #endif // SFML_TCPLISTENER_HPP
117
118
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

TcpSocket.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_TCPSOCKET_HPP
26 #define SFML_TCPSOCKET_HPP
27
29 // Headers
31 #include <SFML/Network/Export.hpp>
32 #include <SFML/Network/Socket.hpp>
33 #include <SFML/System/Time.hpp>
34
35
```

```
36 namespace sf
37 {
38     class TcpListener;
39     class IpAddress;
40     class Packet;
41
46     class SFML_NETWORK_API TcpSocket : public Socket
47     {
48     public:
49
54         TcpSocket();
55
66         unsigned short getLocalPort() const;
67
79         IpAddress getRemoteAddress() const;
80
92         unsigned short getRemotePort() const;
93
111         Status connect(const IpAddress& remoteAddress, unsigned short
Time::Zero);
112
122         void disconnect();
123
140         Status send(const void* data, std::size_t size);
141
156         Status send(const void* data, std::size_t size, std::size_t&
157
174         Status receive(void* data, std::size_t size, std::size_t& re
175
192         Status send(Packet& packet);
193
208         Status receive(Packet& packet);
209
210     private:
211
212         friend class TcpListener;
213
218         struct PendingPacket
219         {
220             PendingPacket();
221
222             UInt32           Size;
223             std::size_t       SizeReceived;
224             std::vector<char> Data;
225         };
226
228         // Member data
230         PendingPacket m_pendingPacket;
231     };
232
233 } // namespace sf
234
```

```
235
236 #endif // SFML_TCPSOCKET_HPP
237
238
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

Text.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_TEXT_HPP
26 #define SFML_TEXT_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Drawable.hpp>
33 #include <SFML/Graphics/Transformable.hpp>
34 #include <SFML/Graphics/Font.hpp>
35 #include <SFML/Graphics/Rect.hpp>
```

```
36 #include <SFML/Graphics/VertexArray.hpp>
37 #include <SFML/System/String.hpp>
38 #include <string>
39 #include <vector>
40
41
42 namespace sf
43 {
48 class SFML_GRAPHICS_API Text : public Drawable, public Transform
49 {
50 public:
51
56     enum Style
57     {
58         Regular      = 0,
59         Bold         = 1 << 0,
60         Italic        = 1 << 1,
61         Underlined    = 1 << 2,
62         StrikeThrough = 1 << 3
63     };
64
71     Text();
72
88     Text(const String& string, const Font& font, unsigned int c
89
109    void setString(const String& string);
110
126    void setFont(const Font& font);
127
145    void setCharacterSize(unsigned int size);
146
159    void setStyle(Uint32 style);
160
177    SFML_DEPRECATED void setColor(const Color& color);
178
191    void setFillColor(const Color& color);
192
203    void setOutlineColor(const Color& color);
204
218    void setOutlineThickness(float thickness);
219
237    const String& getString() const;
238
251    const Font* getFont() const;
252
261    unsigned int getCharacterSize() const;
262
271    Uint32 getStyle() const;
272
285    SFML_DEPRECATED const Color& getColor() const;
286
295    const Color& getFillColor() const;
```

```

296
305     const Color& getOutlineColor() const;
306
315     float getOutlineThickness() const;
316
332     Vector2f findCharacterPos(std::size_t index) const;
333
346     FloatRect getLocalBounds() const;
347
360     FloatRect getGlobalBounds() const;
361
362 private:
363
371     virtual void draw(RenderTarget& target, RenderStates states);
372
380     void ensureGeometryUpdate() const;
381
383 // Member data
385     String          m_string;
386     const Font*      m_font;
387     unsigned int     m_characterSize;
388     Uint32           m_style;
389     Color            m_fillColor;
390     Color            m_outlineColor;
391     float            m_outlineThickness;
392     mutable VertexArray m_vertices;
393     mutable VertexArray m_outlineVertices;
394     mutable FloatRect   m_bounds;
395     mutable bool       m_geometryNeedUpdate;
396 };
397
398 } // namespace sf
399
400
401 #endif // SFML_TEXT_HPP
402
403

```

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Texture.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
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     class SFML_GRAPHICS_API Texture : GlResource
48     {
49     public:
50
55         enum CoordinateType
56         {
57             Normalized,
58             Pixels
59         };
60
61     public:
62
69     Texture();
70
77     Texture(const Texture& copy);
78
83     ~Texture();
84
96     bool create(unsigned int width, unsigned int height);
97
127     bool loadFromFile(const std::string& filename, const IntRect
128
159     bool loadFromMemory(const void* data, std::size_t size, cons
160
190     bool loadFromStream(InputStream& stream, const IntRect& area
191
214     bool loadFromImage(const Image& image, const IntRect& area =
215
222     Vector2u getSize() const;
223
237     Image copyToImage() const;
238
255     void update(const Uint8* pixels);
256
277     void update(const Uint8* pixels, unsigned int width, unsigne
unsigned int y);
278
297     void update(const Image& image);
298
314     void update(const Image& image, unsigned int x, unsigned int
315
334     void update(const Window& window);
335
351     void update(const Window& window, unsigned int x, unsigned i
352
```

```
367     void setSmooth(bool smooth);
368
377     bool isSmooth() const;
378
402     void setSrgb(bool sRgb);
403
412     bool isSrgb() const;
413
436     void setRepeated(bool repeated);
437
446     bool isRepeated() const;
447
471     bool generateMipmap();
472
481     Texture& operator =(const Texture& right);
482
493     unsigned int getNativeHandle() const;
494
526     static void bind(const Texture* texture, CoordinateType coor
527
538     static unsigned int getMaximumSize();
539
540 private:
541
542     friend class RenderTexture;
543     friend class RenderTarget;
544
558     static unsigned int getValidSize(unsigned int size);
559
567     void invalidateMipmap();
568
570     // Member data
572     Vector2u    m_size;
573     Vector2u    m_actualSize;
574     unsigned int m_texture;
575     bool        m_isSmooth;
576     bool        m_sRgb;
577     bool        m_isRepeated;
578     mutable bool m_pixelsFlipped;
579     bool        m_fboAttachment;
580     bool        m_hasMipmap;
581     Uint64      m_cacheId;
582 };
583
584 } // namespace sf
585
586
587 #endif // SFML_TEXTURE_HPP
588
```

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Thread.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_THREAD_HPP
26 #define SFML_THREAD_HPP
27
29 // Headers
31 #include <SFML/System/Export.hpp>
32 #include <SFML/System/NonCopyable.hpp>
33 #include <cstdlib>
34
35
```

```
36 namespace sf
37 {
38 namespace priv
39 {
40     class ThreadImpl;
41     struct ThreadFunc;
42 }
43
48 class SFML_SYSTEM_API Thread : NonCopyable
49 {
50 public:
51
74     template <typename F>
75     Thread(F function);
76
102    template <typename F, typename A>
103    Thread(F function, A argument);
104
125    template <typename C>
126    Thread(void(C::*function)(), C* object);
127
135    ~Thread();
136
146    void launch();
147
159    void wait();
160
172    void terminate();
173
174 private:
175
176     friend class priv::ThreadImpl;
177
184     void run();
185
187     // Member data
189     priv::ThreadImpl* m_impl;
190     priv::ThreadFunc* m_entryPoint;
191 };
192
193 #include <SFML/System/Thread.inl>
194
195 } // namespace sf
196
197 #endif // SFML_THREAD_HPP
198
199
```

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

ThreadLocal.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_THREADLOCAL_HPP
26 #define SFML_THREADLOCAL_HPP
27
29 // Headers
31 #include <SFML/System/Export.hpp>
32 #include <SFML/System/NonCopyable.hpp>
33 #include <cstdlib>
34
35
```

```
36 namespace sf
37 {
38 namespace priv
39 {
40     class ThreadLocalImpl;
41 }
42
47 class SFML_SYSTEM_API ThreadLocal : NonCopyable
48 {
49 public:
50
57     ThreadLocal(void* value = NULL);
58
63     ~ThreadLocal();
64
71     void setValue(void* value);
72
79     void* getValue() const;
80
81 private:
82
84     // Member data
86     priv::ThreadLocalImpl* m_impl;
87 }
88
89 } // namespace sf
90
91
92 #endif // SFML_THREADLOCAL_HPP
93
94
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

ThreadLocalPtr.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_THREADLOCALPTR_HPP
26 #define SFML_THREADLOCALPTR_HPP
27
29 // Headers
31 #include <SFML/System/ThreadLocal.hpp>
32
33
34 namespace sf
35 {
```

```
40 template <typename T>
41 class ThreadLocalPtr : private ThreadLocal
42 {
43 public:
44
45     ThreadLocalPtr(T* value = NULL);
46
47     T& operator *() const;
48
49     T* operator ->() const;
50
51     operator T*() const;
52
53     ThreadLocalPtr<T>& operator =(T* value);
54
55     ThreadLocalPtr<T>& operator =(const ThreadLocalPtr<T>& right);
56 };
57
58 } // namespace sf
59
60 #include <SFML/System/ThreadLocalPtr.inl>
61
62
63 #endif // SFML_THREADLOCALPTR_HPP
64
65
66
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

Time.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_TIME_HPP
26 #define SFML_TIME_HPP
27
29 // Headers
31 #include <SFML/System/Export.hpp>
32
33
34 namespace sf
35 {
```

```
40 class SFML_SYSTEM_API Time
41 {
42 public:
43
44     Time();
45
46     float asSeconds() const;
47
48     Int32 asMilliseconds() const;
49
50     Int64 asMicroseconds() const;
51
52     // Static member data
53     static const Time Zero;
54
55 private:
56
57     friend SFML_SYSTEM_API Time seconds(float);
58     friend SFML_SYSTEM_API Time milliseconds(Int32);
59     friend SFML_SYSTEM_API Time microseconds(Int64);
60
61     explicit Time(Int64 microseconds);
62
63 private:
64
65     // Member data
66     Int64 m_microseconds;
67 };
68
69 SFML_SYSTEM_API Time seconds(float amount);
70
71 SFML_SYSTEM_API Time milliseconds(Int32 amount);
72
73 SFML_SYSTEM_API Time microseconds(Int64 amount);
74
75 SFML_SYSTEM_API bool operator ==(Time left, Time right);
76
77 SFML_SYSTEM_API bool operator !=(Time left, Time right);
78
79 SFML_SYSTEM_API bool operator <(Time left, Time right);
80
81 SFML_SYSTEM_API bool operator >(Time left, Time right);
82
83 SFML_SYSTEM_API bool operator <=(Time left, Time right);
84
85 SFML_SYSTEM_API bool operator >=(Time left, Time right);
86
87 SFML_SYSTEM_API Time operator -(Time right);
88
89 SFML_SYSTEM_API Time operator +(Time left, Time right);
90
91 SFML_SYSTEM_API Time& operator +=(Time& left, Time right);
```

```
257
268 SFML_SYSTEM_API Time operator -(Time left, Time right);
269
280 SFML_SYSTEM_API Time& operator -=(Time& left, Time right);
281
292 SFML_SYSTEM_API Time operator *(Time left, float right);
293
304 SFML_SYSTEM_API Time operator *(Time left, Int64 right);
305
316 SFML_SYSTEM_API Time operator *(float left, Time right);
317
328 SFML_SYSTEM_API Time operator *(Int64 left, Time right);
329
340 SFML_SYSTEM_API Time& operator *=(Time& left, float right);
341
352 SFML_SYSTEM_API Time& operator *=(Time& left, Int64 right);
353
364 SFML_SYSTEM_API Time operator /(Time left, float right);
365
376 SFML_SYSTEM_API Time operator /(Time left, Int64 right);
377
388 SFML_SYSTEM_API Time& operator /=(Time& left, float right);
389
400 SFML_SYSTEM_API Time& operator /=(Time& left, Int64 right);
401
412 SFML_SYSTEM_API float operator /(Time left, Time right);
413
424 SFML_SYSTEM_API Time operator %(Time left, Time right);
425
436 SFML_SYSTEM_API Time& operator %=(Time& left, Time right);
437
438 } // namespace sf
439
440
441 #endif // SFML_TIME_HPP
442
443
```

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Touch.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_TOUCH_HPP
26 #define SFML_TOUCH_HPP
27
29 // Headers
31 #include <SFML/Window/Export.hpp>
32 #include <SFML/System/Vector2.hpp>
33
34
35 namespace sf
```

```
36 {
37 class Window;
38
43 class SFML_WINDOW_API Touch
44 {
45 public:
46
55     static bool isDown(unsigned int finger);
56
68     static Vector2i getPosition(unsigned int finger);
69
82     static Vector2i getPosition(unsigned int finger, const Window& window);
83 }
84
85 } // namespace sf
86
87
88 #endif // SFML_TOUCH_HPP
89
90
```

SFML is licensed under the terms and conditions of the zlib/png license.
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Transform.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_TRANSFORM_HPP
26 #define SFML_TRANSFORM_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Rect.hpp>
33 #include <SFML/System/Vector2.hpp>
34
35
```

```
36 namespace sf
37 {
42 class SFML_GRAPHICS_API Transform
43 {
44 public:
45
52     Transform();
53
68     Transform(float a00, float a01, float a02,
69             float a10, float a11, float a12,
70             float a20, float a21, float a22);
71
87     const float* getMatrix() const;
88
98     Transform getInverse() const;
99
109    Vector2f transformPoint(float x, float y) const;
110
119    Vector2f transformPoint(const Vector2f& point) const;
120
135    FloatRect transformRect(const FloatRect& rectangle) const;
136
149    Transform& combine(const Transform& transform);
150
169    Transform& translate(float x, float y);
170
188    Transform& translate(const Vector2f& offset);
189
207    Transform& rotate(float angle);
208
233    Transform& rotate(float angle, float centerX, float centerY);
234
258    Transform& rotate(float angle, const Vector2f& center);
259
278    Transform& scale(float scaleX, float scaleY);
279
305    Transform& scale(float scaleX, float scaleY, float centerX,
306
324    Transform& scale(const Vector2f& factors);
325
349    Transform& scale(const Vector2f& factors, const Vector2f& c
350
352     // Static member data
354 static const Transform Identity;
355
356 private:
357
359     // Member data
361 float m_matrix[16];
362 };
363
376 SFML_GRAPHICS_API Transform operator *(const Transform& left, co
```

```
377
390 SFML_GRAPHICS_API Transform& operator *=(Transform& left, const
391
404 SFML_GRAPHICS_API Vector2f operator *(const Transform& left, const
405
406 } // namespace sf
407
408
409 #endif // SFML_TRANSFORM_HPP
410
411
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

Transformable.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_TRANSFORMABLE_HPP
26 #define SFML_TRANSFORMABLE_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Transform.hpp>
33
34
35 namespace sf
```

```
36  {
41  class SFML_GRAPHICS_API Transformable
42  {
43  public:
44
49      Transformable();
50
55      virtual ~Transformable();
56
70      void setPosition(float x, float y);
71
84      void setPosition(const Vector2f& position);
85
98      void setRotation(float angle);
99
113     void setScale(float factorX, float factorY);
114
127     void setScale(const Vector2f& factors);
128
145     void setOrigin(float x, float y);
146
162     void setOrigin(const Vector2f& origin);
163
172     const Vector2f& getPosition() const;
173
184     float getRotation() const;
185
194     const Vector2f& getScale() const;
195
204     const Vector2f& getOrigin() const;
205
223     void move(float offsetX, float offsetY);
224
240     void move(const Vector2f& offset);
241
255     void rotate(float angle);
256
274     void scale(float factorX, float factorY);
275
292     void scale(const Vector2f& factor);
293
302     const Transform& getTransform() const;
303
312     const Transform& getInverseTransform() const;
313
314 private:
315
317     // Member data
319     Vector2f          m_origin;
320     Vector2f          m_position;
321     float             m_rotation;
322     Vector2f          m_scale;
```

```
323     mutable Transform m_transform;
324     mutable bool m_transformNeedUpdate;
325     mutable Transform m_inverseTransform;
326     mutable bool m_inverseTransformNeedUpdate;
327 };
328
329 } // namespace sf
330
331
332 #endif // SFML_TRANSFORMABLE_HPP
333
334
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

UdpSocket.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_UDP_SOCKET_HPP
26 #define SFML_UDP_SOCKET_HPP
27
29 // Headers
31 #include <SFML/Network/Export.hpp>
32 #include <SFML/Network/Socket.hpp>
33 #include <SFML/Network/IpAddress.hpp>
34 #include <vector>
35
```

```

36
37 namespace sf
38 {
39 class Packet;
40
45 class SFML_NETWORK_API UdpSocket : public Socket
46 {
47 public:
48
49     // Constants
50     enum
51     {
52         MaxDatagramSize = 65507
53     };
54
55     UdpSocket();
56
57     unsigned short getLocalPort() const;
58
59     Status bind(unsigned short port, const IpAddress& address =
60
61         void unbind();
62
63     Status send(const void* data, std::size_t size, const IpAddr
64         short remotePort);
65
66     Status receive(void* data, std::size_t size, std::size_t& re
67         moteAddress, unsigned short& remotePort);
68
69     Status send(Packet& packet, const IpAddress& remoteAddress,
70
71     Status receive(Packet& packet, IpAddress& remoteAddress, uns
72
73     private:
74
75         // Member data
76         std::vector<char> m_buffer;
77     };
78
79 } // namespace sf
80
81
82 #endif // SFML_UDP_SOCKET_HPP
83
84
85

```

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Utf.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_UTF_HPP
26 #define SFML_UTF_HPP
27
29 // Headers
31 #include <SFML/Config.hpp>
32 #include <algorithm>
33 #include <locale>
34 #include <string>
35 #include <cstdlib>
```

```
36
37
38 namespace sf
39 {
40     template <unsigned int N>
41     class Utf;
42
43     template <>
44     class Utf<8>;
45
46     public:
47
48         template <typename In>
49         static In decode(In begin, In end, Uint32& output, Uint32 replacement);
50
51         template <typename Out>
52         static Out encode(Uint32 input, Out output, Uint8 replacement);
53
54         template <typename In>
55         static In next(In begin, In end);
56
57         template <typename In>
58         static std::size_t count(In begin, In end);
59
60         template <typename In, typename Out>
61         static Out fromAnsi(In begin, In end, Out output, const std::locale& locale);
62
63         template <typename In, typename Out>
64         static Out fromWide(In begin, In end, Out output);
65
66         template <typename In, typename Out>
67         static Out fromLatin1(In begin, In end, Out output);
68
69         template <typename In, typename Out>
70         static Out toAnsi(In begin, In end, Out output, char replacement);
71         std::locale locale = std::locale();
72
73         template <typename In, typename Out>
74         static Out towide(In begin, In end, Out output, wchar_t replacement);
75
76         template <typename In, typename Out>
77         static Out toLatin1(In begin, In end, Out output, char replacement);
78
79         template <typename In, typename Out>
80         static Out toUtf8(In begin, In end, Out output);
81
82         template <typename In, typename Out>
83         static Out toUtf16(In begin, In end, Out output);
84
85         template <typename In, typename Out>
86         static Out toUtf32(In begin, In end, Out output);
```

```
248 };
249
254 template <>
255 class Utf<16>
256 {
257     public:
258
273     template <typename In>
274     static In decode(In begin, In end, Uint32& output, Uint32 re
275
289     template <typename Out>
290     static Out encode(Uint32 input, Out output, Uint16 replaceme
291
304     template <typename In>
305     static In next(In begin, In end);
306
320     template <typename In>
321     static std::size_t count(In begin, In end);
322
337     template <typename In, typename Out>
338     static Out fromAnsi(In begin, In end, Out output, const std:
            std::locale());
339
350     template <typename In, typename Out>
351     static Out fromWide(In begin, In end, Out output);
352
363     template <typename In, typename Out>
364     static Out fromLatin1(In begin, In end, Out output);
365
381     template <typename In, typename Out>
382     static Out toAnsi(In begin, In end, Out output, char replace
            locale = std::locale());
383
395     template <typename In, typename Out>
396     static Out towide(In begin, In end, Out output, wchar_t repl
397
409     template <typename In, typename Out>
410     static Out toLatin1(In begin, In end, Out output, char repla
411
422     template <typename In, typename Out>
423     static Out toUtf8(In begin, In end, Out output);
424
440     template <typename In, typename Out>
441     static Out toUtf16(In begin, In end, Out output);
442
453     template <typename In, typename Out>
454     static Out toUtf32(In begin, In end, Out output);
455 };
456
461 template <>
462 class Utf<32>
463 {
```

```
464 public:
465
481     template <typename In>
482     static In decode(In begin, In end, Uint32& output, Uint32 re
483
498     template <typename Out>
499     static Out encode(Uint32 input, Out output, Uint32 replaceme
500
513     template <typename In>
514     static In next(In begin, In end);
515
528     template <typename In>
529     static std::size_t count(In begin, In end);
530
545     template <typename In, typename Out>
546     static Out fromAnsi(In begin, In end, Out output, const std:
      std::locale());
547
558     template <typename In, typename Out>
559     static Out fromWide(In begin, In end, Out output);
560
571     template <typename In, typename Out>
572     static Out fromLatin1(In begin, In end, Out output);
573
589     template <typename In, typename Out>
590     static Out toAnsi(In begin, In end, Out output, char replace
      locale = std::locale());
591
592     template <typename In, typename Out>
593     static Out towide(In begin, In end, Out output, wchar_t repl
594
595     template <typename In, typename Out>
596     static Out toLatin1(In begin, In end, Out output, char repla
597
598     template <typename In, typename Out>
599     static Out toUtf8(In begin, In end, Out output);
600
601     template <typename In, typename Out>
602     static Out toUtf16(In begin, In end, Out output);
603
604     template <typename In, typename Out>
605     static Out toUtf32(In begin, In end, Out output);
606
607     template <typename In>
608     static Uint32 decodeAnsi(In input, const std::locale& locale
609
610     template <typename In>
611     static Uint32 decodeWide(In input);
612
613     template <typename Out>
614     static Out encodeAnsi(Uint32 codepoint, Out output, char rep
      std::locale& locale = std::locale());
```

```
712
727     template <typename Out>
728     static Out encodeWide(Uint32 codepoint, Out output, wchar_t
729 );
730
731 #include <SFML/System/Utf.inl>
732
733 // Make typedefs to get rid of the template syntax
734 typedef Utf<8> Utf8;
735 typedef Utf<16> Utf16;
736 typedef Utf<32> Utf32;
737
738 } // namespace sf
739
740
741 #endif // SFML_UTF_HPP
742
743
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Vector2.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_VECTOR2_HPP
26 #define SFML_VECTOR2_HPP
27
28
29 namespace sf
30 {
36 template <typename T>
37 class Vector2
38 {
```

```
39 public:
40
47     Vector2();
48
56     Vector2(T X, T Y);
57
69     template <typename U>
70     explicit Vector2(const Vector2<U>& vector);
71
73     // Member data
75     T x;
76     T y;
77 };
78
88 template <typename T>
89 Vector2<T> operator -(const Vector2<T>& right);
90
104 template <typename T>
105 Vector2<T>& operator +=(Vector2<T>& left, const Vector2<T>& rig
106
120 template <typename T>
121 Vector2<T>& operator -=(Vector2<T>& left, const Vector2<T>& rig
122
133 template <typename T>
134 Vector2<T> operator +(const Vector2<T>& left, const Vector2<T>&
135
146 template <typename T>
147 Vector2<T> operator -(const Vector2<T>& left, const Vector2<T>&
148
159 template <typename T>
160 Vector2<T> operator *(const Vector2<T>& left, T right);
161
172 template <typename T>
173 Vector2<T> operator *(T left, const Vector2<T>& right);
174
188 template <typename T>
189 Vector2<T>& operator *=(Vector2<T>& left, T right);
190
201 template <typename T>
202 Vector2<T> operator /(const Vector2<T>& left, T right);
203
217 template <typename T>
218 Vector2<T>& operator /=(Vector2<T>& left, T right);
219
232 template <typename T>
233 bool operator ==(const Vector2<T>& left, const Vector2<T>& right);
234
247 template <typename T>
248 bool operator !=(const Vector2<T>& left, const Vector2<T>& right);
249
250 #include <SFML/System/Vector2.inl>
251
```

```
252 // Define the most common types
253 typedef Vector2<int> Vector2i;
254 typedef Vector2<unsigned int> Vector2u;
255 typedef Vector2<float> Vector2f;
256
257 } // namespace sf
258
259
260 #endif // SFML_VECTOR2_HPP
261
262
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Vector3.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_VECTOR3_HPP
26 #define SFML_VECTOR3_HPP
27
28
29 namespace sf
30 {
36 template <typename T>
37 class Vector3
38 {
```

```
39 public:
40
47     Vector3();
48
57     Vector3(T X, T Y, T Z);
58
70     template <typename U>
71     explicit Vector3(const Vector3<U>& vector);
72
74     // Member data
76     T x;
77     T y;
78     T z;
79 };
80
90 template <typename T>
91 Vector3<T> operator -(const Vector3<T>& left);
92
106 template <typename T>
107 Vector3<T>& operator +=(Vector3<T>& left, const Vector3<T>& rig
108
122 template <typename T>
123 Vector3<T>& operator -=(Vector3<T>& left, const Vector3<T>& rig
124
135 template <typename T>
136 Vector3<T> operator +(const Vector3<T>& left, const Vector3<T>&
137
148 template <typename T>
149 Vector3<T> operator -(const Vector3<T>& left, const Vector3<T>&
150
161 template <typename T>
162 Vector3<T> operator *(const Vector3<T>& left, T right);
163
174 template <typename T>
175 Vector3<T> operator *(T left, const Vector3<T>& right);
176
190 template <typename T>
191 Vector3<T>& operator *=(Vector3<T>& left, T right);
192
203 template <typename T>
204 Vector3<T> operator / (const Vector3<T>& left, T right);
205
219 template <typename T>
220 Vector3<T>& operator /=(Vector3<T>& left, T right);
221
234 template <typename T>
235 bool operator ==(const Vector3<T>& left, const Vector3<T>& right);
236
249 template <typename T>
250 bool operator !=(const Vector3<T>& left, const Vector3<T>& right);
251
252 #include <SFML/System/Vector3.inl>
```

```
253 // Define the most common types
254 typedef Vector3<int> Vector3i;
255 typedef Vector3<float> Vector3f;
256
257 } // namespace sf
258
259
260
261 #endif // SFML_VECTOR3_HPP
262
263
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Vertex.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_VERTEX_HPP
26 #define SFML_VERTEX_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Color.hpp>
33 #include <SFML/System/Vector2.hpp>
34
35
```

```
36 namespace sf
37 {
42 class SFML_GRAPHICS_API Vertex
43 {
44 public:
45     Vertex();
50     Vertex(const Vector2f& thePosition);
51     Vertex(const Vector2f& thePosition, const Color& theColor);
52     Vertex(const Vector2f& thePosition, const Vector2f& theTexC
53     Vertex(const Vector2f& thePosition, const Color& theColor,
54         // Member data
55     Vector2f position;
56     Color color;
57     Vector2f texCoords;
58 };
59 }
60 } // namespace sf
61
62 #endif // SFML_VERTEX_HPP
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

VertexArray.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_VERTEXARRAY_HPP
26 #define SFML_VERTEXARRAY_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Vertex.hpp>
33 #include <SFML/Graphics/PrimitiveType.hpp>
34 #include <SFML/Graphics/Rect.hpp>
35 #include <SFML/Graphics/Drawable.hpp>
```

```
36 #include <vector>
37
38
39 namespace sf
40 {
41     class SFML_GRAPHICS_API VertexArray : public Drawable
42     {
43     public:
44
45         VertexArray();
46
47         explicit VertexArray(PrimitiveType type, std::size_t vertexCount);
48
49         std::size_t getVertexCount() const;
50
51         Vertex& operator [] (std::size_t index);
52
53         const Vertex& operator [] (std::size_t index) const;
54
55         void clear();
56
57         void resize(std::size_t vertexCount);
58
59         void append(const Vertex& vertex);
60
61         void setPrimitiveType(PrimitiveType type);
62
63         PrimitiveType getPrimitiveType() const;
64
65         FloatRect getBounds() const;
66
67     private:
68
69         virtual void draw(RenderTarget& target, RenderStates states) const;
70
71     private:
72
73         // Member data
74         std::vector<Vertex> m_vertices;
75         PrimitiveType          m_primitiveType;
76     };
77
78 } // namespace sf
79
80
81 #endif // SFML_VERTEXARRAY_HPP
82
83
84
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

VideoMode.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_VIDEOMODE_HPP
26 #define SFML_VIDEOMODE_HPP
27
29 // Headers
31 #include <SFML/Window/Export.hpp>
32 #include <vector>
33
34
35 namespace sf
```

```
36  {
41  class SFML_WINDOW_API VideoMode
42  {
43  public:
44
51      VideoMode();
52
61      VideoMode(unsigned int modeWidth, unsigned int modeHeight, u
32);
62
69      static VideoMode getDesktopMode();
70
85      static const std::vector<VideoMode>& getFullscreenModes();
86
97      bool isValid() const;
98
100     // Member data
102     unsigned int width;
103     unsigned int height;
104     unsigned int bitsPerPixel;
105 };
106
117 SFML_WINDOW_API bool operator ==(const VideoMode& left, const Vi
118
129 SFML_WINDOW_API bool operator !=(const VideoMode& left, const Vi
130
141 SFML_WINDOW_API bool operator <(const VideoMode& left, const Vi
142
153 SFML_WINDOW_API bool operator >(const VideoMode& left, const Vi
154
165 SFML_WINDOW_API bool operator <=(const VideoMode& left, const Vi
166
177 SFML_WINDOW_API bool operator >=(const VideoMode& left, const Vi
178
179 } // namespace sf
180
181
182 #endif // SFML_VIDEO_MODE_HPP
183
184
```

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | Classes | **Files** |

| File List

View.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_VIEW_HPP
26 #define SFML_VIEW_HPP
27
29 // Headers
31 #include <SFML/Graphics/Export.hpp>
32 #include <SFML/Graphics/Rect.hpp>
33 #include <SFML/Graphics/Transform.hpp>
34 #include <SFML/System/Vector2.hpp>
35
```

```
36
37 namespace sf
38 {
39 class SFML_GRAPHICS_API View
40 {
41 public:
42
43     View();
44
45     explicit View(const FloatRect& rectangle);
46
47     View(const Vector2f& center, const Vector2f& size);
48
49     void setCenter(float x, float y);
50
51     void setCenter(const Vector2f& center);
52
53     void setSize(float width, float height);
54
55     void setSize(const Vector2f& size);
56
57     void setRotation(float angle);
58
59     void setViewport(const FloatRect& viewport);
60
61     void reset(const FloatRect& rectangle);
62
63     const Vector2f& getCenter() const;
64
65     const Vector2f& getSize() const;
66
67     float getRotation() const;
68
69     const FloatRect& getViewport() const;
70
71     void move(float offsetX, float offsetY);
72
73     void move(const Vector2f& offset);
74
75     void rotate(float angle);
76
77     void zoom(float factor);
78
79     const Transform& getTransform() const;
80
81     const Transform& getInverseTransform() const;
82
83 private:
84
85     // Member data
86     Vector2f          m_center;
87     Vector2f          m_size;
```

```
274     float           m_rotation;
275     FloatRect       m_viewport;
276     mutable Transform m_transform;
277     mutable Transform m_inverseTransform;
278     mutable bool    m_transformUpdated;
279     mutable bool    m_invTransformUpdated;
280 };
281
282 } // namespace sf
283
284
285 #endif // SFML_VIEW_HPP
286
287
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

Window/Window.hpp

```
1 //
3 // SFML - Simple and Fast Multimedia Library
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)
5 //
6 // This software is provided 'as-is', without any express or imp
7 // In no event will the authors be held liable for any damages a
software.
8 //
9 // Permission is granted to anyone to use this software for any
10 // including commercial applications, and to alter it and redist
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 n
17 //
18 // 2. Altered source versions must be plainly marked as such,
19 //     and must not be misrepresented as being the original softw
20 //
21 // 3. This notice may not be removed or altered from any source
22 //
24
25 #ifndef SFML_WINDOW_HPP
26 #define SFML_WINDOW_HPP
27
29 // Headers
31 #include <SFML/Window/Export.hpp>
32 #include <SFML/Window/ContextSettings.hpp>
33 #include <SFML/Window/VideoMode.hpp>
34 #include <SFML/Window/WindowHandle.hpp>
35 #include <SFML/Window/WindowStyle.hpp>
```

```
36 #include <SFML/Window/GlResource.hpp>
37 #include <SFML/System/Clock.hpp>
38 #include <SFML/System/Vector2.hpp>
39 #include <SFML/System/NonCopyable.hpp>
40 #include <SFML/System/String.hpp>
41
42
43 namespace sf
44 {
45     namespace priv
46     {
47         class GlContext;
48         class WindowImpl;
49     }
50
51     class Event;
52
57     class SFML_WINDOW_API Window : GlResource, NonCopyable
58     {
59     public:
60
68         Window();
69
89         Window(VideoMode mode, const String& title, Uint32 style =
    ContextSettings& settings = ContextSettings());
90
105        explicit Window(WindowHandle handle, const ContextSettings&
106
113        virtual ~Window();
114
132        void create(VideoMode mode, const String& title, Uint32 sty
    ContextSettings& settings = ContextSettings());
133
149        void create(WindowHandle handle, const ContextSettings& set
150
161        void close();
162
173        bool isOpen() const;
174
186        const ContextSettings& getSettings() const;
187
211        bool pollEvent(Event& event);
212
238        bool waitEvent(Event& event);
239
248        Vector2i getPosition() const;
249
262        void setPosition(const Vector2i& position);
263
275        Vector2u getSize() const;
276
285        void setSize(const Vector2u& size);
```

```
286     void setTitle(const String& title);
295
296     void setIcon(unsigned int width, unsigned int height, const
314
315     void setVisible(bool visible);
324
325     void setVerticalSyncEnabled(bool enabled);
339
340     void setMouseCursorVisible(bool visible);
349
350     void setMouseCursorGrabbed(bool grabbed);
362
363     void setKeyRepeatEnabled(bool enabled);
376
377     void setFramerateLimit(unsigned int limit);
393
394     void setJoystickThreshold(float threshold);
406
407     bool setActive(bool active = true) const;
424
425     void requestFocus();
440
441     bool hasFocus() const;
453
454     void display();
463
464     WindowHandle getSystemHandle() const;
477
478 protected:
479
480     virtual void onCreate();
490
498     virtual void onResize();
499
500 private:
501
514     bool filterEvent(const Event& event);
515
520     void initialize();
521
523     // Member data
525     priv::WindowImpl* m_impl;
526     priv::GlContext* m_context;
527     Clock m_clock;
528     Time m_frameTimeLimit;
529     Vector2u m_size;
530 };
531
532 } // namespace sf
533
534
```

```
535 #endif // SFML_WINDOW_HPP
536
537
```

SFML is licensed under the terms and conditions of the [zlib/png license](#).

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#).

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

Window.hpp

```
1 //  
3 // SFML - Simple and Fast Multimedia Library  
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)  
5 //  
6 // This software is provided 'as-is', without any express or imp  
7 // In no event will the authors be held liable for any damages a  
software.  
8 //  
9 // Permission is granted to anyone to use this software for any  
10 // including commercial applications, and to alter it and redist  
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 n  
17 //  
18 // 2. Altered source versions must be plainly marked as such,  
19 //     and must not be misrepresented as being the original softw  
20 //  
21 // 3. This notice may not be removed or altered from any source  
22 //  
24  
25 #ifndef SFML_SFML_WINDOW_HPP  
26 #define SFML_SFML_WINDOW_HPP  
27  
29 // Headers  
31  
32 #include <SFML/System.hpp>  
33 #include <SFML/Window/Context.hpp>  
34 #include <SFML/Window/ContextSettings.hpp>  
35 #include <SFML/Window/Event.hpp>
```

```
36 #include <SFML/Window/Joystick.hpp>
37 #include <SFML/Window/Keyboard.hpp>
38 #include <SFML/Window/Mouse.hpp>
39 #include <SFML/Window/Sensor.hpp>
40 #include <SFML/Window/Touch.hpp>
41 #include <SFML/Window/VideoMode.hpp>
42 #include <SFML/Window/Window.hpp>
43 #include <SFML/Window/WindowHandle.hpp>
44 #include <SFML/Window/WindowStyle.hpp>
45
46
47
48 #endif // SFML_SFML_WINDOW_HPP
49
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

WindowHandle.hpp

```
1 //  
3 // SFML - Simple and Fast Multimedia Library  
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)  
5 //  
6 // This software is provided 'as-is', without any express or imp  
7 // In no event will the authors be held liable for any damages a  
software.  
8 //  
9 // Permission is granted to anyone to use this software for any  
10 // including commercial applications, and to alter it and redist  
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 n  
17 //  
18 // 2. Altered source versions must be plainly marked as such,  
19 //     and must not be misrepresented as being the original softw  
20 //  
21 // 3. This notice may not be removed or altered from any source  
22 //  
24  
25 #ifndef SFML_WINDOWHANDLE_HPP  
26 #define SFML_WINDOWHANDLE_HPP  
27  
29 // Headers  
31 #include <SFML/Config.hpp>  
32  
33 // Windows' HWND is a typedef on struct HWND__*  
34 #if defined(SFML_SYSTEM_WINDOWS)  
    struct HWND__;
```

```

36 #endif
37
38 namespace sf
39 {
40 #if defined(SFML_SYSTEM_WINDOWS)
41
42     // Window handle is HWND (HWND__*) on Windows
43     typedef HWND__* WindowHandle;
44
45 #elif defined(SFML_SYSTEM_LINUX) || defined(SFML_SYSTEM_FREEBSD)
46
47     // Window handle is Window (unsigned long) on Unix - X11
48     typedef unsigned long WindowHandle;
49
50 #elif defined(SFML_SYSTEM_MACOS)
51
52     // Window handle is NSWindow or NSView (void*) on Mac OS X -
53     typedef void* WindowHandle;
54
55 #elif defined(SFML_SYSTEM_IOS)
56
57     // Window handle is UIWindow (void*) on iOS - UIKit
58     typedef void* WindowHandle;
59
60 #elif defined(SFML_SYSTEM_ANDROID)
61
62     // Window handle is ANativeWindow* (void*) on Android
63     typedef void* WindowHandle;
64
65 #elif defined(SFML_DOXYGEN)
66
67     // Define typedef symbol so that Doxygen can attach some doc
68     typedef "platform-specific" WindowHandle;
69
70 #endif
71
72 } // namespace sf
73
74
75#endif // SFML_WINDOWHANDLE_HPP
76

```

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[File List](#)

WindowStyle.hpp

```
1 //  
3 // SFML - Simple and Fast Multimedia Library  
4 // Copyright (C) 2007-2017 Laurent Gomila (laurent@sfml-dev.org)  
5 //  
6 // This software is provided 'as-is', without any express or imp  
7 // In no event will the authors be held liable for any damages a  
software.  
8 //  
9 // Permission is granted to anyone to use this software for any  
10 // including commercial applications, and to alter it and redist  
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 n  
17 //  
18 // 2. Altered source versions must be plainly marked as such,  
19 //     and must not be misrepresented as being the original softw  
20 //  
21 // 3. This notice may not be removed or altered from any source  
22 //  
24  
25 #ifndef SFML_WINDOWSTYLE_HPP  
26 #define SFML_WINDOWSTYLE_HPP  
27  
28  
29 namespace sf  
30 {  
31 namespace Style  
32 {  
38     enum
```

```
39     {
40         None      = 0,
41         Titlebar  = 1 << 0,
42         Resize    = 1 << 1,
43         Close     = 1 << 2,
44         Fullscreen = 1 << 3,
45
46         Default = Titlebar | Resize | Close
47     };
48 }
49
50 } // namespace sf
51
52
53 #endif // SFML_WINDOWSTYLE_HPP
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)

Related Pages

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

Related Pages

Here is a list of all related documentation pages:

[Deprecated List](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::AIResource Member List

This is the complete list of members for [sf::AIResource](#), including all inher

[AIResource\(\)](#) sf::AIResource protected

[~AIResource\(\)](#) sf::AIResource protected

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::InputSoundFile Member List

This is the complete list of members for [sf::InputSoundFile](#), including all in

[getChannelCount\(\)](#) const

[getDuration\(\)](#) const

[getSampleCount\(\)](#) const

[getSampleRate\(\)](#) const

[InputSoundFile\(\)](#)

[NonCopyable\(\)](#)

[openForWriting\(const std::string &filename, unsigned int channelCount, u](#)

[openFromFile\(const std::string &filename\)](#)

[openFromMemory\(const void *data, std::size_t sizeInBytes\)](#)

[openFromStream\(InputStream &stream\)](#)

[read\(Int16 *samples, Uint64 maxCount\)](#)

[seek\(Uint64 sampleOffset\)](#)

[seek\(Time timeOffset\)](#)

[~InputSoundFile\(\)](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Listener Member List

This is the complete list of members for [sf::Listener](#), including all inherited

getDirection()	sf::Listener	static
getGlobalVolume()	sf::Listener	static
getPosition()	sf::Listener	static
getUpVector()	sf::Listener	static
setDirection(float x, float y, float z)	sf::Listener	static
setDirection(const Vector3f &direction)	sf::Listener	static
setGlobalVolume(float volume)	sf::Listener	static
setPosition(float x, float y, float z)	sf::Listener	static
setPosition(const Vector3f &position)	sf::Listener	static
setUpVector(float x, float y, float z)	sf::Listener	static
setUpVector(const Vector3f &upVector)	sf::Listener	static

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Music Member List

This is the complete list of members for [sf::Music](#), including all inherited members.

[getAttenuation\(\) const](#)

[getChannelCount\(\) const](#)

[getDuration\(\) const](#)

[getLoop\(\) const](#)

[getMinDistance\(\) const](#)

[getPitch\(\) const](#)

[getPlayingOffset\(\) const](#)

[getPosition\(\) const](#)

[getSampleRate\(\) const](#)

[getStatus\(\) const](#)

[getVolume\(\) const](#)

[sf::SoundStream::initialize\(unsigned int channelCount, unsigned int sampleRate\) const](#)

[isRelativeToListener\(\) const](#)

[m_source](#)

`Music()`

`onGetData(Chunk &data)`

`onSeek(Time timeOffset)`

`openFromFile(const std::string &filename)`

`openFromMemory(const void *data, std::size_t sizeInBytes)`

`openFromStream(InputStream &stream)`

`operator=(const SoundSource &right)`

`pause()`

Paused enum value

`play()`

Playing enum value

`setAttenuation(float attenuation)`

`setLoop(bool loop)`

`setMinDistance(float distance)`

`setPitch(float pitch)`

`setPlayingOffset(Time timeOffset)`

`setPosition(float x, float y, float z)`

`setPosition(const Vector3f &position)`

`setRelativeToListener(bool relative)`

`setVolume(float volume)`

`SoundSource(const SoundSource ©)`

`SoundSource()`

`SoundStream()`

Status enum name

`stop()`

Stopped enum value

`~Music()`

`~SoundSource()`

`~SoundStream()`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::OutputSoundFile Member List

This is the complete list of members for `sf::OutputSoundFile`, including all

`NonCopyable()`

`openFromFile(const std::string &filename, unsigned int sampleRate, unsigned int channels)`

`OutputSoundFile()`

`write(const Int16 *samples, Uint64 count)`

`~OutputSoundFile()`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Sound Member List

This is the complete list of members for `sf::Sound`, including all inherited members.

<code>getAttenuation() const</code>	<code>sf::SoundSource</code>
<code>getBuffer() const</code>	<code>sf::Sound</code>
<code>getLoop() const</code>	<code>sf::Sound</code>
<code>getMinDistance() const</code>	<code>sf::SoundSource</code>
<code>getPitch() const</code>	<code>sf::SoundSource</code>
<code>getPlayingOffset() const</code>	<code>sf::Sound</code>
<code>getPosition() const</code>	<code>sf::SoundSource</code>
<code>getStatus() const</code>	<code>sf::Sound</code>
<code>getVolume() const</code>	<code>sf::SoundSource</code>
<code>isRelativeToListener() const</code>	<code>sf::SoundSource</code>
<code>m_source</code>	<code>sf::SoundSource</code>
<code>operator=(const Sound &right)</code>	<code>sf::Sound</code>
<code>sf::SoundSource::operator=(const SoundSource &right)</code>	<code>sf::SoundSource</code>
<code>pause()</code>	<code>sf::Sound</code>

Paused enum value	sf::SoundSource
play()	sf::Sound
Playing enum value	sf::SoundSource
resetBuffer()	sf::Sound
setAttenuation(float attenuation)	sf::SoundSource
setBuffer(const SoundBuffer &buffer)	sf::Sound
setLoop(bool loop)	sf::Sound
setMinDistance(float distance)	sf::SoundSource
setPitch(float pitch)	sf::SoundSource
setPlayingOffset(Time timeOffset)	sf::Sound
setPosition(float x, float y, float z)	sf::SoundSource
setPosition(const Vector3f &position)	sf::SoundSource
setRelativeToListener(bool relative)	sf::SoundSource
setVolume(float volume)	sf::SoundSource
Sound()	sf::Sound
Sound(const SoundBuffer &buffer)	sf::Sound
Sound(const Sound ©)	sf::Sound
SoundSource(const SoundSource ©)	sf::SoundSource
SoundSource()	sf::SoundSource
Status enum name	sf::SoundSource
stop()	sf::Sound
Stopped enum value	sf::SoundSource
-Sound()	sf::Sound
-SoundSource()	sf::SoundSource

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

sf::SoundBuffer Member List

This is the complete list of members for `sf::SoundBuffer`, including all inhe

`AIResource()`

`getChannelCount() const`

`getDuration() const`

`getSampleCount() const`

`getSampleRate() const`

`getSamples() const`

`loadFromFile(const std::string &filename)`

`loadFromMemory(const void *data, std::size_t sizeInBytes)`

`loadFromSamples(const Int16 *samples, Uint64 sampleCount, unsigned int channelCount)`

`loadFromStream(InputStream &stream)`

`operator=(const SoundBuffer &right)`

`saveToFile(const std::string &filename) const`

Sound (defined in `sf::SoundBuffer`)

`SoundBuffer()`

`SoundBuffer(const SoundBuffer ©)`

`~AIResource()`

`~SoundBuffer()`

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::SoundBufferRecorder Member List

This is the complete list of members for `sf::SoundBufferRecorder`, including all inherited members.

<code>getAvailableDevices()</code>	<code>sf::SoundBufferRecorder</code>
<code>getBuffer() const</code>	<code>sf::SoundBufferRecorder</code>
<code>getChannelCount() const</code>	<code>sf::SoundBufferRecorder</code>
<code>getDefaultDevice()</code>	<code>sf::SoundBufferRecorder</code>
<code>getDevice() const</code>	<code>sf::SoundBufferRecorder</code>
<code>getSampleRate() const</code>	<code>sf::SoundBufferRecorder</code>
<code>isAvailable()</code>	<code>sf::SoundBufferRecorder</code>
<code>onProcessSamples(const Int16 *samples, std::size_t sampleCount)</code>	<code>sf::SoundBufferRecorder</code>
<code>onStart()</code>	<code>sf::SoundBufferRecorder</code>
<code>onStop()</code>	<code>sf::SoundBufferRecorder</code>
<code>setChannelCount(unsigned int channelCount)</code>	<code>sf::SoundBufferRecorder</code>
<code>setDevice(const std::string &name)</code>	<code>sf::SoundBufferRecorder</code>
<code>setProcessingInterval(Time interval)</code>	<code>sf::SoundBufferRecorder</code>
<code>SoundRecorder()</code>	<code>sf::SoundBufferRecorder</code>

start(unsigned int sampleRate=44100)	sf::SoundRecorder
stop()	sf::SoundRecorder
~SoundBufferRecorder()	sf::SoundRecorder
~SoundRecorder()	sf::SoundRecorder

SFML is licensed under the terms and conditions of the zlib/png license.
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::SoundFileFactory Member List

This is the complete list of members for [sf::SoundFileFactory](#), including all inherited members.

createReaderFromFilename(const std::string &filename)	sf::SoundFileFactory
createReaderFromMemory(const void *data, std::size_t sizeInBytes)	sf::SoundFileFactory
createReaderFromStream(InputStream &stream)	sf::SoundFileFactory
createWriterFromFilename(const std::string &filename)	sf::SoundFileFactory
registerReader()	sf::SoundFileFactory
registerWriter()	sf::SoundFileFactory
unregisterReader()	sf::SoundFileFactory
unregisterWriter()	sf::SoundFileFactory

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen 1.8.13

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::SoundFileReader Member List

This is the complete list of members for `sf::SoundFileReader`, including all inherited members.

<code>open(InputStream &stream, Info &info)=0</code>	<code>sf::SoundFileReader</code>	<small>pure virtual</small>
<code>read(Int16 *samples, Uint64 maxCount)=0</code>	<code>sf::SoundFileReader</code>	<small>pure virtual</small>
<code>seek(Uint64 sampleOffset)=0</code>	<code>sf::SoundFileReader</code>	<small>pure virtual</small>
<code>-SoundFileReader()</code>	<code>sf::SoundFileReader</code>	<small>inline virtual</small>

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::SoundFileReader::Info Member List

This is the complete list of members for `sf::SoundFileReader::Info`, includi

`channelCount` `sf::SoundFileReader::Info`

`sampleCount` `sf::SoundFileReader::Info`

`sampleRate` `sf::SoundFileReader::Info`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::SoundFileWriter Member List

This is the complete list of members for `sf::SoundFileWriter`, including all i

```
open(const std::string &filename, unsigned int sampleRate, unsigned int c
write(const Int16 *samples, Uint64 count)=0
~SoundFileWriter()
```

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::SoundRecorder Member List

This is the complete list of members for `sf::SoundRecorder`, including all inherited members.

<code>AIResource()</code>	<code>sf::</code>
<code>getAvailableDevices()</code>	<code>sf::</code>
<code>getChannelCount() const</code>	<code>sf::</code>
<code>getDefaultDevice()</code>	<code>sf::</code>
<code>getDevice() const</code>	<code>sf::</code>
<code>getSampleRate() const</code>	<code>sf::</code>
<code>isAvailable()</code>	<code>sf::</code>
<code>onProcessSamples(const Int16 *samples, std::size_t sampleCount)=0</code>	<code>sf::</code>
<code>onStart()</code>	<code>sf::</code>
<code>onStop()</code>	<code>sf::</code>
<code>setChannelCount(unsigned int channelCount)</code>	<code>sf::</code>
<code>setDevice(const std::string &name)</code>	<code>sf::</code>
<code>setProcessingInterval(Time interval)</code>	<code>sf::</code>
<code>SoundRecorder()</code>	<code>sf::</code>

start(unsigned int sampleRate=44100)	sf:
stop()	sf:
~AIResource()	sf:
~SoundRecorder()	sf:

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::SoundSource Member List

This is the complete list of members for `sf::SoundSource`, including all inh

<code>AIResource()</code>	<code>sf::AIResource</code>	private
<code>getAttenuation() const</code>	<code>sf::SoundSource</code>	
<code>getMinDistance() const</code>	<code>sf::SoundSource</code>	
<code>getPitch() const</code>	<code>sf::SoundSource</code>	
<code>getPosition() const</code>	<code>sf::SoundSource</code>	
<code>getStatus() const</code>	<code>sf::SoundSource</code>	protected
<code>getVolume() const</code>	<code>sf::SoundSource</code>	
<code>isRelativeToListener() const</code>	<code>sf::SoundSource</code>	
<code>m_source</code>	<code>sf::SoundSource</code>	protected
<code>operator=(const SoundSource &right)</code>	<code>sf::SoundSource</code>	
<code>Paused</code> enum value	<code>sf::SoundSource</code>	
<code>Playing</code> enum value	<code>sf::SoundSource</code>	
<code>setAttenuation(float attenuation)</code>	<code>sf::SoundSource</code>	
<code>setMinDistance(float distance)</code>	<code>sf::SoundSource</code>	

setPitch(float pitch)	sf::SoundSource
setPosition(float x, float y, float z)	sf::SoundSource
setPosition(const Vector3f &position)	sf::SoundSource
setRelativeToListener(bool relative)	sf::SoundSource
setVolume(float volume)	sf::SoundSource
SoundSource(const SoundSource ©)	sf::SoundSource
SoundSource()	sf::SoundSource protected
Status enum name	sf::SoundSource
Stopped enum value	sf::SoundSource
~AIResource()	sf::AIResource private
~SoundSource()	sf::SoundSource virtual

SFML is licensed under the terms and conditions of the zlib/png
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::SoundStream Member List

This is the complete list of members for [sf::SoundStream](#), including all inh

<code>getAttenuation() const</code>	sf::Sound
<code>getChannelCount() const</code>	sf::Sound
<code>getLoop() const</code>	sf::Sound
<code>getMinDistance() const</code>	sf::Sound
<code>getPitch() const</code>	sf::Sound
<code>getPlayingOffset() const</code>	sf::Sound
<code>getPosition() const</code>	sf::Sound
<code>getSampleRate() const</code>	sf::Sound
<code>getStatus() const</code>	sf::Sound
<code>getVolume() const</code>	sf::Sound
<code>initialize(unsigned int channelCount, unsigned int sampleRate)</code>	sf::Sound
<code>isRelativeToListener() const</code>	sf::Sound
<code>m_source</code>	sf::Sound
<code>onGetData(Chunk &data)=0</code>	sf::Sound

onSeek(Time timeOffset)=0	sf::Sound
operator=(const SoundSource &right)	sf::Sound
pause()	sf::Sound
Paused enum value	sf::Sound
play()	sf::Sound
Playing enum value	sf::Sound
setAttenuation(float attenuation)	sf::Sound
setLoop(bool loop)	sf::Sound
setMinDistance(float distance)	sf::Sound
setPitch(float pitch)	sf::Sound
setPlayingOffset(Time timeOffset)	sf::Sound
setPosition(float x, float y, float z)	sf::Sound
setPosition(const Vector3f &position)	sf::Sound
setRelativeToListener(bool relative)	sf::Sound
setVolume(float volume)	sf::Sound
SoundSource(const SoundSource ©)	sf::Sound
SoundSource()	sf::Sound
SoundStream()	sf::Sound
Status enum name	sf::Sound
stop()	sf::Sound
Stopped enum value	sf::Sound
~SoundSource()	sf::Sound
~SoundStream()	sf::Sound

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::SoundStream::Chunk Member List

This is the complete list of members for `sf::SoundStream::Chunk`, includin

`sampleCount` `sf::SoundStream::Chunk`

`samples` `sf::SoundStream::Chunk`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::BlendMode Member List

This is the complete list of members for [sf::BlendMode](#), including all inherited members.

[Add](#) enum value

[alphaDstFactor](#)

[alphaEquation](#)

[alphaSrcFactor](#)

[BlendMode\(\)](#)

[BlendMode\(Factor sourceFactor, Factor destinationFactor, Equation blendEquation\)](#)

[BlendMode\(Factor colorSourceFactor, Factor colorDestinationFactor, Equation colorEquation\)](#)

[colorDstFactor](#)

[colorEquation](#)

[colorSrcFactor](#)

[DstAlpha](#) enum value

[DstColor](#) enum value

[Equation](#) enum name

[Factor](#) enum name

`One` enum value

`OneMinusDstAlpha` enum value

`OneMinusDstColor` enum value

`OneMinusSrcAlpha` enum value

`OneMinusSrcColor` enum value

`operator!=`(const BlendMode &left, const BlendMode &right)

`operator==`(const BlendMode &left, const BlendMode &right)

`ReverseSubtract` enum value

`SrcAlpha` enum value

`SrcColor` enum value

`Subtract` enum value

`Zero` enum value

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::CircleShape Member List

This is the complete list of members for `sf::CircleShape`, including all inhe

<code>CircleShape(float radius=0, std::size_t pointCount=30)</code>	<code>sf::CircleShape</code>
<code>getFillColor() const</code>	<code>sf::Shape</code>
<code>getGlobalBounds() const</code>	<code>sf::Shape</code>
<code>getInverseTransform() const</code>	<code>sf::Transformable</code>
<code>getLocalBounds() const</code>	<code>sf::Shape</code>
<code>getOrigin() const</code>	<code>sf::Transformable</code>
<code>getOutlineColor() const</code>	<code>sf::Shape</code>
<code>getOutlineThickness() const</code>	<code>sf::Shape</code>
<code>getPoint(std::size_t index) const</code>	<code>sf::CircleShape</code>
<code>getPointCount() const</code>	<code>sf::CircleShape</code>
<code>getPosition() const</code>	<code>sf::Transformable</code>
<code>getRadius() const</code>	<code>sf::CircleShape</code>
<code>getRotation() const</code>	<code>sf::Transformable</code>
<code>getScale() const</code>	<code>sf::Transformable</code>

getTexture() const	sf::Shape
getTextureRect() const	sf::Shape
getTransform() const	sf::Transformable
move(float offsetX, float offsetY)	sf::Transformable
move(const Vector2f &offset)	sf::Transformable
rotate(float angle)	sf::Transformable
scale(float factorX, float factorY)	sf::Transformable
scale(const Vector2f &factor)	sf::Transformable
setFillColor(const Color &color)	sf::Shape
setOrigin(float x, float y)	sf::Transformable
setOrigin(const Vector2f &origin)	sf::Transformable
setOutlineColor(const Color &color)	sf::Shape
setOutlineThickness(float thickness)	sf::Shape
setPointCount(std::size_t count)	sf::CircleShape
setPosition(float x, float y)	sf::Transformable
setPosition(const Vector2f &position)	sf::Transformable
setRadius(float radius)	sf::CircleShape
setRotation(float angle)	sf::Transformable
setScale(float factorX, float factorY)	sf::Transformable
setScale(const Vector2f &factors)	sf::Transformable
setTexture(const Texture *texture, bool resetRect=false)	sf::Shape
setTextureRect(const IntRect &rect)	sf::Shape
Shape()	sf::Shape
Transformable()	sf::Transformable
update()	sf::Shape
~Drawable()	sf::Drawable

`-Shape()`

`sf::Shape`

`-Transformable()`

`sf::Transformable`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Color Member List

This is the complete list of members for `sf::Color`, including all inherited members.

a	<code>sf::Color</code>	
b	<code>sf::Color</code>	
Black	<code>sf::Color</code>	<code>static</code>
Blue	<code>sf::Color</code>	<code>static</code>
Color()	<code>sf::Color</code>	
<code>Color(Uint8 red, Uint8 green, Uint8 blue, Uint8 alpha=255)</code>	<code>sf::Color</code>	
<code>Color(Uint32 color)</code>	<code>sf::Color</code>	<code>explicit</code>
Cyan	<code>sf::Color</code>	<code>static</code>
g	<code>sf::Color</code>	
Green	<code>sf::Color</code>	<code>static</code>
Magenta	<code>sf::Color</code>	<code>static</code>
<code>operator!=(const Color &left, const Color &right)</code>	<code>sf::Color</code>	<code>related</code>
<code>operator*(const Color &left, const Color &right)</code>	<code>sf::Color</code>	<code>related</code>
<code>operator*=(Color &left, const Color &right)</code>	<code>sf::Color</code>	<code>related</code>

operator+(const Color &left, const Color &right)	sf::Color	related
operator+=(Color &left, const Color &right)	sf::Color	related
operator-(const Color &left, const Color &right)	sf::Color	related
operator-=(Color &left, const Color &right)	sf::Color	related
operator==(const Color &left, const Color &right)	sf::Color	related
r	sf::Color	
Red	sf::Color	static
tolInteger() const	sf::Color	
Transparent	sf::Color	static
White	sf::Color	static
Yellow	sf::Color	static

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::ConvexShape Member List

This is the complete list of members for `sf::ConvexShape`, including all inherited members.

<code>ConvexShape(std::size_t pointCount=0)</code>	<code>sf::ConvexShape</code>
<code>getFillColor() const</code>	<code>sf::Shape</code>
<code>getGlobalBounds() const</code>	<code>sf::Shape</code>
<code>getInverseTransform() const</code>	<code>sf::Transformable</code>
<code>getLocalBounds() const</code>	<code>sf::Shape</code>
<code>getOrigin() const</code>	<code>sf::Transformable</code>
<code>getOutlineColor() const</code>	<code>sf::Shape</code>
<code>getOutlineThickness() const</code>	<code>sf::Shape</code>
<code>getPoint(std::size_t index) const</code>	<code>sf::ConvexShape</code>
<code>getPointCount() const</code>	<code>sf::ConvexShape</code>
<code>getPosition() const</code>	<code>sf::Transformable</code>
<code>getRotation() const</code>	<code>sf::Transformable</code>
<code>getScale() const</code>	<code>sf::Transformable</code>
<code>getTexture() const</code>	<code>sf::Shape</code>

getTextureRect() const	sf::Shape
getTransform() const	sf::Transformable
move(float offsetX, float offsetY)	sf::Transformable
move(const Vector2f &offset)	sf::Transformable
rotate(float angle)	sf::Transformable
scale(float factorX, float factorY)	sf::Transformable
scale(const Vector2f &factor)	sf::Transformable
setFillColor(const Color &color)	sf::Shape
setOrigin(float x, float y)	sf::Transformable
setOrigin(const Vector2f &origin)	sf::Transformable
setOutlineColor(const Color &color)	sf::Shape
setOutlineThickness(float thickness)	sf::Shape
setPoint(std::size_t index, const Vector2f &point)	sf::ConvexShape
setPointCount(std::size_t count)	sf::ConvexShape
setPosition(float x, float y)	sf::Transformable
setPosition(const Vector2f &position)	sf::Transformable
setRotation(float angle)	sf::Transformable
setScale(float factorX, float factorY)	sf::Transformable
setScale(const Vector2f &factors)	sf::Transformable
setTexture(const Texture *texture, bool resetRect=false)	sf::Shape
setTextureRect(const IntRect ▭)	sf::Shape
Shape()	sf::Shape
Transformable()	sf::Transformable
update()	sf::Shape
~Drawable()	sf::Drawable
~Shape()	sf::Shape

`-Transformable()`

`sf::Transformable`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Drawable Member List

This is the complete list of members for [sf::Drawable](#), including all inheritance.

draw(RenderTarget &target, RenderStates states) const =0	sf::Drawable
RenderTarget (defined in sf::Drawable)	sf::Drawable
~Drawable()	sf::Drawable

SFML is licensed under the terms and conditions of the zlib/pn.

Copyright © Laurent Gomila :: Documentation generated by doxygen.

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Font Member List

This is the complete list of members for `sf::Font`, including all inherited members.

`Font()`

`Font(const Font ©)`

`getGlyph(Uint32 codePoint, unsigned int characterSize, bool bold, float outlineWidth)`

`getInfo() const`

`getKerning(Uint32 first, Uint32 second, unsigned int characterSize) const`

`getLineSpacing(unsigned int characterSize) const`

`getTexture(unsigned int characterSize) const`

`getUnderlinePosition(unsigned int characterSize) const`

`getUnderlineThickness(unsigned int characterSize) const`

`loadFromFile(const std::string &filename)`

`loadFromMemory(const void *data, std::size_t sizeInBytes)`

`loadFromStream(InputStream &stream)`

`operator=(const Font &right)`

`~Font()`

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Font::Info Member List

This is the complete list of members for [sf::Font::Info](#), including all inheritance

[family](#) [sf::Font::Info](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Glyph Member List

This is the complete list of members for [sf::Glyph](#), including all inherited members.

[advance](#) sf::Glyph

[bounds](#) sf::Glyph

[Glyph\(\)](#) sf::Glyph inline

[textureRect](#) sf::Glyph

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Image Member List

This is the complete list of members for `sf::Image`, including all inherited members.

`copy(const Image &source, unsigned int destX, unsigned int destY, const`
`create(unsigned int width, unsigned int height, const Color &color=Color(0, 0, 0))`
`create(unsigned int width, unsigned int height, const Uint8 *pixels)`
`createMaskFromColor(const Color &color, Uint8 alpha=0)`
`flipHorizontally()`
`flipVertically()`
`getPixel(unsigned int x, unsigned int y) const`
`getPixelsPtr() const`
`getSize() const`
`Image()`
`loadFromFile(const std::string &filename)`
`loadFromMemory(const void *data, std::size_t size)`
`loadFromStream(InputStream &stream)`
`saveToFile(const std::string &filename) const`

```
setPixel(unsigned int x, unsigned int y, const Color &color)
```

```
~Image()
```

SFML is licensed under the terms and conditions of the zlib/png

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

`sf::Rect< T >` Member List

This is the complete list of members for `sf::Rect< T >`, including all inherited members.

<code>contains(T x, T y) const</code>	<code>sf::</code>
<code>contains(const Vector2< T > &point) const</code>	<code>sf::</code>
<code>height</code>	<code>sf::</code>
<code>intersects(const Rect< T > &rectangle) const</code>	<code>sf::</code>
<code>intersects(const Rect< T > &rectangle, Rect< T > &intersection) const</code>	<code>sf::</code>
<code>left</code>	<code>sf::</code>
<code>operator!=(const Rect< T > &left, const Rect< T > &right)</code>	<code>sf::</code>
<code>operator==(const Rect< T > &left, const Rect< T > &right)</code>	<code>sf::</code>
<code>Rect()</code>	<code>sf::</code>
<code>Rect(T rectLeft, T rectTop, T rectWidth, T rectHeight)</code>	<code>sf::</code>
<code>Rect(const Vector2< T > &position, const Vector2< T > &size)</code>	<code>sf::</code>
<code>Rect(const Rect< U > &rectangle)</code>	<code>sf::</code>
<code>top</code>	<code>sf::</code>
<code>width</code>	<code>sf::</code>

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::RectangleShape Member List

This is the complete list of members for `sf::RectangleShape`, including all

<code>getFillColor() const</code>	<code>sf::Shape</code>
<code>getGlobalBounds() const</code>	<code>sf::Shape</code>
<code>getInverseTransform() const</code>	<code>sf::Transformable</code>
<code>getLocalBounds() const</code>	<code>sf::Shape</code>
<code>getOrigin() const</code>	<code>sf::Transformable</code>
<code>getOutlineColor() const</code>	<code>sf::Shape</code>
<code>getOutlineThickness() const</code>	<code>sf::Shape</code>
<code>getPoint(std::size_t index) const</code>	<code>sf::RectangleShape</code>
<code>getPointCount() const</code>	<code>sf::RectangleShape</code>
<code>getPosition() const</code>	<code>sf::Transformable</code>
<code>getRotation() const</code>	<code>sf::Transformable</code>
<code>getScale() const</code>	<code>sf::Transformable</code>
<code>getSize() const</code>	<code>sf::RectangleShape</code>
<code>getTexture() const</code>	<code>sf::Shape</code>

getTextureRect() const	sf::Shape
getTransform() const	sf::Transformable
move(float offsetX, float offsetY)	sf::Transformable
move(const Vector2f &offset)	sf::Transformable
RectangleShape(const Vector2f &size=Vector2f(0, 0))	sf::RectangleShape
rotate(float angle)	sf::Transformable
scale(float factorX, float factorY)	sf::Transformable
scale(const Vector2f &factor)	sf::Transformable
setFillColor(const Color &color)	sf::Shape
setOrigin(float x, float y)	sf::Transformable
setOrigin(const Vector2f &origin)	sf::Transformable
setOutlineColor(const Color &color)	sf::Shape
setOutlineThickness(float thickness)	sf::Shape
setPosition(float x, float y)	sf::Transformable
setPosition(const Vector2f &position)	sf::Transformable
setRotation(float angle)	sf::Transformable
setScale(float factorX, float factorY)	sf::Transformable
setScale(const Vector2f &factors)	sf::Transformable
setSize(const Vector2f &size)	sf::RectangleShape
setTexture(const Texture *texture, bool resetRect=false)	sf::Shape
setTextureRect(const IntRect ▭)	sf::Shape
Shape()	sf::Shape
Transformable()	sf::Transformable
update()	sf::Shape
~Drawable()	sf::Drawable
~Shape()	sf::Shape

`-Transformable()`

`sf::Transformable`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::RenderStates Member List

This is the complete list of members for `sf::RenderStates`, including all inh

blendMode

Default

RenderStates()

RenderStates(const BlendMode &theBlendMode)

RenderStates(const Transform &theTransform)

RenderStates(const Texture *theTexture)

RenderStates(const Shader *theShader)

RenderStates(const BlendMode &theBlendMode, const Transform &theTr

shader

texture

transform

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::RenderTarget Member List

This is the complete list of members for [sf::RenderTarget](#), including all inh

[clear\(const Color &color=Color\(0, 0, 0, 255\)\)](#)

[draw\(const Drawable &drawable, const RenderStates &states=RenderSta](#)

[draw\(const Vertex *vertices, std::size_t vertexCount, PrimitiveType type, c](#)

[getDefaultView\(\) const](#)

[getSize\(\) const =0](#)

[getView\(\) const](#)

[getViewport\(const View &view\) const](#)

[initialize\(\)](#)

[mapCoordsToPixel\(const Vector2f &point\) const](#)

[mapCoordsToPixel\(const Vector2f &point, const View &view\) const](#)

[mapPixelToCoords\(const Vector2i &point\) const](#)

[mapPixelToCoords\(const Vector2i &point, const View &view\) const](#)

[NonCopyable\(\)](#)

[popGLStates\(\)](#)

`pushGLStates()`

`RenderTarget()`

`resetGLStates()`

`setView(const View &view)`

`~RenderTarget()`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::RenderTexture Member List

This is the complete list of members for [sf::RenderTexture](#), including all in

[clear\(const Color &color=Color\(0, 0, 0, 255\)\)](#)

[create\(unsigned int width, unsigned int height, bool depthBuffer=false\)](#)

[display\(\)](#)

[draw\(const Drawable &drawable, const RenderStates &states=RenderSta](#)

[draw\(const Vertex *vertices, std::size_t vertexCount, PrimitiveType type, c](#)

[generateMipmap\(\)](#)

[getDefaultView\(\) const](#)

[getSize\(\) const](#)

[getTexture\(\) const](#)

[getView\(\) const](#)

[getViewport\(const View &view\) const](#)

[initialize\(\)](#)

[isRepeated\(\) const](#)

[isSmooth\(\) const](#)

```
mapCoordsToPixel(const Vector2f &point) const  
mapCoordsToPixel(const Vector2f &point, const View &view) const  
mapPixelToCoords(const Vector2i &point) const  
mapPixelToCoords(const Vector2i &point, const View &view) const  
popGLStates()  
pushGLStates()  
RenderTarget()  
RenderTexture()  
resetGLStates()  
setActive(bool active=true)  
setRepeated(bool repeated)  
setSmooth(bool smooth)  
setView(const View &view)  
~RenderTarget()  
~RenderTexture()
```

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::RenderWindow Member List

This is the complete list of members for [sf::RenderWindow](#), including all ir

[capture\(\) const](#)

[clear\(const Color &color=Color\(0, 0, 0, 255\)\)](#)

[close\(\)](#)

[create\(VideoMode mode, const String &title, Uint32 style=Style::Default, const ContextSettings &settings=ContextSettings\(\)\)](#)

[create\(WindowHandle handle, const ContextSettings &settings=ContextSettings\(\)\)](#)

[display\(\)](#)

[draw\(const Drawable &drawable, const RenderStates &states=RenderStates\(\)\)](#)

[draw\(const Vertex *vertices, std::size_t vertexCount, PrimitiveType type, const RenderStates &states=RenderStates\(\)\)](#)

[getDefaultView\(\) const](#)

[getPosition\(\) const](#)

[getSettings\(\) const](#)

[getSize\(\) const](#)

[getSystemHandle\(\) const](#)

[getView\(\) const](#)

```
getViewport(const View &view) const
hasFocus() const
initialize()
isOpen() const
mapCoordsToPixel(const Vector2f &point) const
mapCoordsToPixel(const Vector2f &point, const View &view) const
mapPixelToCoords(const Vector2i &point) const
mapPixelToCoords(const Vector2i &point, const View &view) const
NonCopyable()
onCreate()
onResize()
pollEvent(Event &event)
popGLStates()
pushGLStates()
RenderTarget()
RenderWindow()
RenderWindow(VideoMode mode, const String &title, Uint32 style=Style::)
RenderWindow(WindowHandle handle, const ContextSettings &settings=
requestFocus()
resetGLStates()
setActive(bool active=true) const
setFramerateLimit(unsigned int limit)
setIcon(unsigned int width, unsigned int height, const Uint8 *pixels)
setJoystickThreshold(float threshold)
setKeyRepeatEnabled(bool enabled)
setMouseCursorGrabbed(bool grabbed)
```

```
setMouseCursorVisible(bool visible)
setPosition(const Vector2i &position)
setSize(const Vector2u &size)
setTitle(const String &title)
setVerticalSyncEnabled(bool enabled)
setView(const View &view)
setVisible(bool visible)
waitEvent(Event &event)
Window()
Window(VideoMode mode, const String &title, Uint32 style=Style::Default)
Window(WindowHandle handle, const ContextSettings &settings=Context
~RenderTarget()
~RenderWindow()
~Window()
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Shader Member List

This is the complete list of members for [sf::Shader](#), including all inherited

[bind\(const Shader *shader\)](#)

[CurrentTexture](#)

[ensureGIContext\(\)](#)

[Fragment](#) enum value

[Geometry](#) enum value

[getNativeHandle\(\) const](#)

[GIResource\(\)](#)

[isAvailable\(\)](#)

[isGeometryAvailable\(\)](#)

[loadFromFile\(const std::string &filename, Type type\)](#)

[loadFromFile\(const std::string &vertexShaderFilename, const std::string &](#)

[loadFromFile\(const std::string &vertexShaderFilename, const std::string &](#)

[loadFromMemory\(const std::string &shader, Type type\)](#)

[loadFromMemory\(const std::string &vertexShader, const std::string &fragme](#)

```
loadFromMemory(const std::string &vertexShader, const std::string &geo  
loadFromStream(InputStream &stream, Type type)  
loadFromStream(InputStream &vertexShaderStream, InputStream &fragr  
loadFromStream(InputStream &vertexShaderStream, InputStream &geom  
NonCopyable()  
setParameter(const std::string &name, float x)  
setParameter(const std::string &name, float x, float y)  
setParameter(const std::string &name, float x, float y, float z)  
setParameter(const std::string &name, float x, float y, float z, float w)  
setParameter(const std::string &name, const Vector2f &vector)  
setParameter(const std::string &name, const Vector3f &vector)  
setParameter(const std::string &name, const Color &color)  
setParameter(const std::string &name, const Transform &transform)  
setParameter(const std::string &name, const Texture &texture)  
setParameter(const std::string &name, CurrentTextureType)  
setUniform(const std::string &name, float x)  
setUniform(const std::string &name, const Gsl::Vec2 &vector)  
setUniform(const std::string &name, const Gsl::Vec3 &vector)  
setUniform(const std::string &name, const Gsl::Vec4 &vector)  
setUniform(const std::string &name, int x)  
setUniform(const std::string &name, const Gsl::Ivec2 &vector)  
setUniform(const std::string &name, const Gsl::Ivec3 &vector)  
setUniform(const std::string &name, const Gsl::Ivec4 &vector)  
setUniform(const std::string &name, bool x)  
setUniform(const std::string &name, const Gsl::Bvec2 &vector)  
setUniform(const std::string &name, const Gsl::Bvec3 &vector)
```

```
setUniform(const std::string &name, const Gsl::Bvec4 &vector)
setUniform(const std::string &name, const Gsl::Mat3 &matrix)
setUniform(const std::string &name, const Gsl::Mat4 &matrix)
setUniform(const std::string &name, const Texture &texture)
setUniform(const std::string &name, CurrentTextureType)
setUniformArray(const std::string &name, const float *scalarArray, std::size_t size)
setUniformArray(const std::string &name, const Gsl::Vec2 *vectorArray, std::size_t size)
setUniformArray(const std::string &name, const Gsl::Vec3 *vectorArray, std::size_t size)
setUniformArray(const std::string &name, const Gsl::Vec4 *vectorArray, std::size_t size)
setUniformArray(const std::string &name, const Gsl::Mat3 *matrixArray, std::size_t size)
setUniformArray(const std::string &name, const Gsl::Mat4 *matrixArray, std::size_t size)
Shader()
Type enum name
Vertex enum value
~GIResource()
~Shader()
```

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Shape Member List

This is the complete list of members for [sf::Shape](#), including all inherited members.

getFillColor() const	sf::Shape
getGlobalBounds() const	sf::Shape
getInverseTransform() const	sf::Transformable
getLocalBounds() const	sf::Shape
getOrigin() const	sf::Transformable
getOutlineColor() const	sf::Shape
getOutlineThickness() const	sf::Shape
getPoint(std::size_t index) const =0	sf::Shape
getPointCount() const =0	sf::Shape
getPosition() const	sf::Transformable
getRotation() const	sf::Transformable
getScale() const	sf::Transformable
getTexture() const	sf::Shape
getTextureRect() const	sf::Shape

<code>getTransform() const</code>	<code>sf::Transformable</code>
<code>move(float offsetX, float offsetY)</code>	<code>sf::Transformable</code>
<code>move(const Vector2f &offset)</code>	<code>sf::Transformable</code>
<code>rotate(float angle)</code>	<code>sf::Transformable</code>
<code>scale(float factorX, float factorY)</code>	<code>sf::Transformable</code>
<code>scale(const Vector2f &factor)</code>	<code>sf::Transformable</code>
<code>setFillColor(const Color &color)</code>	<code>sf::Shape</code>
<code>setOrigin(float x, float y)</code>	<code>sf::Transformable</code>
<code>setOrigin(const Vector2f &origin)</code>	<code>sf::Transformable</code>
<code>setOutlineColor(const Color &color)</code>	<code>sf::Shape</code>
<code>setOutlineThickness(float thickness)</code>	<code>sf::Shape</code>
<code>setPosition(float x, float y)</code>	<code>sf::Transformable</code>
<code>setPosition(const Vector2f &position)</code>	<code>sf::Transformable</code>
<code>setRotation(float angle)</code>	<code>sf::Transformable</code>
<code>setScale(float factorX, float factorY)</code>	<code>sf::Transformable</code>
<code>setScale(const Vector2f &factors)</code>	<code>sf::Transformable</code>
<code>setTexture(const Texture *texture, bool resetRect=false)</code>	<code>sf::Shape</code>
<code>setTextureRect(const IntRect ▭)</code>	<code>sf::Shape</code>
<code>Shape()</code>	<code>sf::Shape</code>
<code>Transformable()</code>	<code>sf::Transformable</code>
<code>update()</code>	<code>sf::Shape</code>
<code>~Drawable()</code>	<code>sf::Drawable</code>
<code>~Shape()</code>	<code>sf::Shape</code>
<code>~Transformable()</code>	<code>sf::Transformable</code>

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Sprite Member List

This is the complete list of members for [sf::Sprite](#), including all inherited members.

getColor() const	sf::Sprite
getGlobalBounds() const	sf::Sprite
getInverseTransform() const	sf::Transformable
getLocalBounds() const	sf::Sprite
getOrigin() const	sf::Transformable
getPosition() const	sf::Transformable
getRotation() const	sf::Transformable
getScale() const	sf::Transformable
getTexture() const	sf::Sprite
getTextureRect() const	sf::Sprite
getTransform() const	sf::Transformable
move(float offsetX, float offsetY)	sf::Transformable
move(const Vector2f &offset)	sf::Transformable
rotate(float angle)	sf::Transformable

<code>scale(float factorX, float factorY)</code>	<code>sf::Transformab</code>
<code>scale(const Vector2f &factor)</code>	<code>sf::Transformab</code>
<code>setColor(const Color &color)</code>	<code>sf::Sprite</code>
<code>setOrigin(float x, float y)</code>	<code>sf::Transformab</code>
<code>setOrigin(const Vector2f &origin)</code>	<code>sf::Transformab</code>
<code>setPosition(float x, float y)</code>	<code>sf::Transformab</code>
<code>setPosition(const Vector2f &position)</code>	<code>sf::Transformab</code>
<code>setRotation(float angle)</code>	<code>sf::Transformab</code>
<code>setScale(float factorX, float factorY)</code>	<code>sf::Transformab</code>
<code>setScale(const Vector2f &factors)</code>	<code>sf::Transformab</code>
<code>setTexture(const Texture &texture, bool resetRect=false)</code>	<code>sf::Sprite</code>
<code>setTextureRect(const IntRect &rectangle)</code>	<code>sf::Sprite</code>
<code>Sprite()</code>	<code>sf::Sprite</code>
<code>Sprite(const Texture &texture)</code>	<code>sf::Sprite</code>
<code>Sprite(const Texture &texture, const IntRect &rectangle)</code>	<code>sf::Sprite</code>
<code>Transformable()</code>	<code>sf::Transformab</code>
<code>~Drawable()</code>	<code>sf::Drawable</code>
<code>~Transformable()</code>	<code>sf::Transformab</code>

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Text Member List

This is the complete list of members for [sf::Text](#), including all inherited me

Bold enum value

[findCharacterPos\(std::size_t index\) const](#)

[getCharacterSize\(\) const](#)

[getColor\(\) const](#)

[getFillColor\(\) const](#)

[getFont\(\) const](#)

[getGlobalBounds\(\) const](#)

[getInverseTransform\(\) const](#)

[getLocalBounds\(\) const](#)

[getOrigin\(\) const](#)

[getOutlineColor\(\) const](#)

[getOutlineThickness\(\) const](#)

[getPosition\(\) const](#)

[getRotation\(\) const](#)

`getScale() const`
`getString() const`
`getStyle() const`
`getTransform() const`
`Italic` enum value
`move(float offsetX, float offsetY)`
`move(const Vector2f &offset)`
Regular enum value
`rotate(float angle)`
`scale(float factorX, float factorY)`
`scale(const Vector2f &factor)`
`setCharacterSize(unsigned int size)`
`setColor(const Color &color)`
`setFillColor(const Color &color)`
`setFont(const Font &font)`
`setOrigin(float x, float y)`
`setOrigin(const Vector2f &origin)`
`setOutlineColor(const Color &color)`
`setOutlineThickness(float thickness)`
`setPosition(float x, float y)`
`setPosition(const Vector2f &position)`
`setRotation(float angle)`
`setScale(float factorX, float factorY)`
`setScale(const Vector2f &factors)`
`setString(const String &string)`
`setStyle(UInt32 style)`

StrikeThrough enum value

Style enum name

Text()

Text(const String &string, const Font &font, unsigned int characterSize=30)

Transformable()

Underlined enum value

~Drawable()

~Transformable()

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by **doxygen**

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Texture Member List

This is the complete list of members for [sf::Texture](#), including all inherited

[bind\(const Texture *texture, CoordinateType coordinateType=Normalized\)](#)
[CoordinateType enum name](#)
[copyToImage\(\) const](#)
[create\(unsigned int width, unsigned int height\)](#)
[ensureGIContext\(\)](#)
[generateMipmap\(\)](#)
[getMaximumSize\(\)](#)
[getNativeHandle\(\) const](#)
[getSize\(\) const](#)
[GIResource\(\)](#)
[isRepeated\(\) const](#)
[isSmooth\(\) const](#)
[isSrgb\(\) const](#)
[loadFromFile\(const std::string &filename, const IntRect &area=IntRect\(\)\)](#)

```
loadFromImage(const Image &image, const IntRect &area=IntRect())
loadFromMemory(const void *data, std::size_t size, const IntRect &area=
loadFromStream(InputStream &stream, const IntRect &area=IntRect())
Normalized enum value
operator=(const Texture &right)
Pixels enum value
RenderTarget (defined in sf::Texture)
RenderTexture (defined in sf::Texture)
setRepeated(bool repeated)
setSmooth(bool smooth)
setSrgb(bool sRgb)
Texture()
Texture(const Texture &copy)
update(const Uint8 *pixels)
update(const Uint8 *pixels, unsigned int width, unsigned int height, unsigr
update(const Image &image)
update(const Image &image, unsigned int x, unsigned int y)
update(const Window &window)
update(const Window &window, unsigned int x, unsigned int y)
~GIResource()
~Texture()
```

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Transform Member List

This is the complete list of members for `sf::Transform`, including all inherited members.

`combine(const Transform &transform)`

`getInverse() const`

`getMatrix() const`

`Identity`

`operator*(const Transform &left, const Transform &right)`

`operator*(const Transform &left, const Vector2f &right)`

`operator*=(Transform &left, const Transform &right)`

`rotate(float angle)`

`rotate(float angle, float centerX, float centerY)`

`rotate(float angle, const Vector2f ¢er)`

`scale(float scaleX, float scaleY)`

`scale(float scaleX, float scaleY, float centerX, float centerY)`

`scale(const Vector2f &factors)`

`scale(const Vector2f &factors, const Vector2f ¢er)`

Transform()

```
Transform(float a00, float a01, float a02, float a10, float a11, float a12, float a13)
transformPoint(float x, float y) const
transformPoint(const Vector2f &point) const
transformRect(const FloatRect &rectangle) const
translate(float x, float y)
translate(const Vector2f &offset)
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

sf::Transformable Member List

This is the complete list of members for **sf::Transformable**, including all inherited members.

<code>getInverseTransform() const</code>	<code>sf::Transformable</code>
<code>getOrigin() const</code>	<code>sf::Transformable</code>
<code>getPosition() const</code>	<code>sf::Transformable</code>
<code>getRotation() const</code>	<code>sf::Transformable</code>
<code>getScale() const</code>	<code>sf::Transformable</code>
<code>getTransform() const</code>	<code>sf::Transformable</code>
<code>move(float offsetX, float offsetY)</code>	<code>sf::Transformable</code>
<code>move(const Vector2f &offset)</code>	<code>sf::Transformable</code>
<code>rotate(float angle)</code>	<code>sf::Transformable</code>
<code>scale(float factorX, float factorY)</code>	<code>sf::Transformable</code>
<code>scale(const Vector2f &factor)</code>	<code>sf::Transformable</code>
<code>setOrigin(float x, float y)</code>	<code>sf::Transformable</code>
<code>setOrigin(const Vector2f &origin)</code>	<code>sf::Transformable</code>
<code>setPosition(float x, float y)</code>	<code>sf::Transformable</code>

<code>setPosition(const Vector2f &position)</code>	<code>sf::Transformable</code>
<code>setRotation(float angle)</code>	<code>sf::Transformable</code>
<code>setScale(float factorX, float factorY)</code>	<code>sf::Transformable</code>
<code>setScale(const Vector2f &factors)</code>	<code>sf::Transformable</code>
<code>Transformable()</code>	<code>sf::Transformable</code>
<code>~Transformable()</code>	<code>sf::Transformable</code> <small>virtual</small>

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

`sf::Vertex` Member List

This is the complete list of members for `sf::Vertex`, including all inherited members.

`color`

`position`

`texCoords`

`Vertex()`

`Vertex(const Vector2f &thePosition)`

`Vertex(const Vector2f &thePosition, const Color &theColor)`

`Vertex(const Vector2f &thePosition, const Vector2f &theTexCoords)`

`Vertex(const Vector2f &thePosition, const Color &theColor, const Vector2f &theTexCoords)`

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::VertexArray Member List

This is the complete list of members for `sf::VertexArray`, including all inher

<code>append(const Vertex &vertex)</code>	<code>sf::VertexArray</code>
<code>clear()</code>	<code>sf::VertexArray</code>
<code>getBounds() const</code>	<code>sf::VertexArray</code>
<code>getPrimitiveType() const</code>	<code>sf::VertexArray</code>
<code>getVertexCount() const</code>	<code>sf::VertexArray</code>
<code>operator[](std::size_t index)</code>	<code>sf::VertexArray</code>
<code>operator[](std::size_t index) const</code>	<code>sf::VertexArray</code>
<code>resize(std::size_t vertexCount)</code>	<code>sf::VertexArray</code>
<code>setPrimitiveType(PrimitiveType type)</code>	<code>sf::VertexArray</code>
<code>VertexArray()</code>	<code>sf::VertexArray</code>
<code>VertexArray(PrimitiveType type, std::size_t vertexCount=0)</code>	<code>sf::VertexArray</code>
<code>~Drawable()</code>	<code>sf::Drawable</code>

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::View Member List

This is the complete list of members for `sf::View`, including all inherited members.

<code>getCenter() const</code>	<code>sf::View</code>
<code>getInverseTransform() const</code>	<code>sf::View</code>
<code>getRotation() const</code>	<code>sf::View</code>
<code>getSize() const</code>	<code>sf::View</code>
<code>getTransform() const</code>	<code>sf::View</code>
<code>getViewport() const</code>	<code>sf::View</code>
<code>move(float offsetX, float offsetY)</code>	<code>sf::View</code>
<code>move(const Vector2f &offset)</code>	<code>sf::View</code>
<code>reset(const FloatRect &rectangle)</code>	<code>sf::View</code>
<code>rotate(float angle)</code>	<code>sf::View</code>
<code>setCenter(float x, float y)</code>	<code>sf::View</code>
<code>setCenter(const Vector2f &center)</code>	<code>sf::View</code>
<code>setRotation(float angle)</code>	<code>sf::View</code>
<code>setSize(float width, float height)</code>	<code>sf::View</code>

setSize(const Vector2f &size)	sf::View
setViewport(const FloatRect &viewport)	sf::View
View()	sf::View
View(const FloatRect &rectangle)	sf::View
View(const Vector2f ¢er, const Vector2f &size)	sf::View
zoom(float factor)	sf::View

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

sf::Ftp Member List

This is the complete list of members for `sf::Ftp`, including all inherited members.

`Ascii` enum value

`Binary` enum value

`changeDirectory(const std::string &directory)`

`connect(const IpAddress &server, unsigned short port=21, Time timeout=5s)`

`createDirectory(const std::string &name)`

DataChannel (defined in `sf::Ftp`)

`deleteDirectory(const std::string &name)`

`deleteFile(const std::string &name)`

`disconnect()`

`download(const std::string &remoteFile, const std::string &localPath, TransferMode mode=TransferMode::Binary)`

`Ebcdic` enum value

`getDirectoryListing(const std::string &directory="")`

`getWorkingDirectory()`

`keepAlive()`

```
login()
login(const std::string &name, const std::string &password)
NonCopyable()
parentDirectory()
renameFile(const std::string &file, const std::string &newName)
sendCommand(const std::string &command, const std::string &parameters)
TransferMode enum name
upload(const std::string &localFile, const std::string &remotePath, TransferMode mode)
~Ftp()
```

SFML is licensed under the terms and conditions of the zlib/png license.
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Ftp::DirectoryResponse Member List

This is the complete list of members for [sf::Ftp::DirectoryResponse](#), includ

[BadCommandSequence](#) enum value

[ClosingConnection](#) enum value

[ClosingDataConnection](#) enum value

[CommandNotImplemented](#) enum value

[CommandUnknown](#) enum value

[ConnectionClosed](#) enum value

[ConnectionFailed](#) enum value

[DataConnectionAlreadyOpened](#) enum value

[DataConnectionOpened](#) enum value

[DataConnectionUnavailable](#) enum value

[DirectoryOk](#) enum value

[DirectoryResponse](#)(const Response &response)

[DirectoryStatus](#) enum value

[EnteringPassiveMode](#) enum value

FileActionAborted enum value

FileActionOk enum value

FilenameNotAllowed enum value

FileStatus enum value

FileUnavailable enum value

getDirectory() const

getMessage() const

getStatus() const

HelpMessage enum value

InsufficientStorageSpace enum value

InvalidFile enum value

InvalidResponse enum value

isOk() const

LocalError enum value

LoggedIn enum value

NeedAccountToLogin enum value

NeedAccountToStore enum value

NeedInformation enum value

NeedPassword enum value

NotEnoughMemory enum value

NotLoggedIn enum value

Ok enum value

OpeningDataConnection enum value

PageTypeUnknown enum value

ParameterNotImplemented enum value

ParametersUnknown enum value

PointlessCommand enum value

Response(Status code=InvalidResponse, const std::string &message="")

RestartMarkerReply enum value

ServiceReady enum value

ServiceReadySoon enum value

ServiceUnavailable enum value

Status enum name

SystemStatus enum value

SystemType enum value

TransferAborted enum value

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Ftp::ListingResponse Member List

This is the complete list of members for [sf::Ftp::ListingResponse](#), including all inherited members.

[BadCommandSequence](#) enum value

[ClosingConnection](#) enum value

[ClosingDataConnection](#) enum value

[CommandNotImplemented](#) enum value

[CommandUnknown](#) enum value

[ConnectionClosed](#) enum value

[ConnectionFailed](#) enum value

[DataConnectionAlreadyOpened](#) enum value

[DataConnectionOpened](#) enum value

[DataConnectionUnavailable](#) enum value

[DirectoryOk](#) enum value

[DirectoryStatus](#) enum value

[EnteringPassiveMode](#) enum value

[FileActionAborted](#) enum value

FileActionOk enum value

FilenameNotAllowed enum value

FileStatus enum value

FileUnavailable enum value

getListing() const

getMessage() const

getStatus() const

HelpMessage enum value

InsufficientStorageSpace enum value

InvalidFile enum value

InvalidResponse enum value

isOk() const

ListingResponse(const Response &response, const std::string &data)

LocalError enum value

LoggedIn enum value

NeedAccountToLogin enum value

NeedAccountToStore enum value

NeedInformation enum value

NeedPassword enum value

NotEnoughMemory enum value

NotLoggedIn enum value

Ok enum value

OpeningDataConnection enum value

PageTypeUnknown enum value

ParameterNotImplemented enum value

ParametersUnknown enum value

PointlessCommand enum value

Response(Status code=InvalidResponse, const std::string &message="")

RestartMarkerReply enum value

ServiceReady enum value

ServiceReadySoon enum value

ServiceUnavailable enum value

Status enum name

SystemStatus enum value

SystemType enum value

TransferAborted enum value

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Ftp::Response Member List

This is the complete list of members for [sf::Ftp::Response](#), including all inheritance-based members.

[BadCommandSequence](#) enum value

[ClosingConnection](#) enum value

[ClosingDataConnection](#) enum value

[CommandNotImplemented](#) enum value

[CommandUnknown](#) enum value

[ConnectionClosed](#) enum value

[ConnectionFailed](#) enum value

[DataConnectionAlreadyOpened](#) enum value

[DataConnectionOpened](#) enum value

[DataConnectionUnavailable](#) enum value

[DirectoryOk](#) enum value

[DirectoryStatus](#) enum value

[EnteringPassiveMode](#) enum value

[FileActionAborted](#) enum value

FileActionOk enum value

FilenameNotAllowed enum value

FileStatus enum value

FileUnavailable enum value

getMessage() const

getStatus() const

HelpMessage enum value

InsufficientStorageSpace enum value

InvalidFile enum value

InvalidResponse enum value

isOk() const

LocalError enum value

LoggedIn enum value

NeedAccountToLogIn enum value

NeedAccountToStore enum value

NeedInformation enum value

NeedPassword enum value

NotEnoughMemory enum value

NotLoggedIn enum value

Ok enum value

OpeningDataConnection enum value

PageTypeUnknown enum value

ParameterNotImplemented enum value

ParametersUnknown enum value

PointlessCommand enum value

Response(Status code=InvalidResponse, const std::string &message="")

`RestartMarkerReply` enum value

`ServiceReady` enum value

`ServiceReadySoon` enum value

`ServiceUnavailable` enum value

`Status` enum name

`SystemStatus` enum value

`SystemType` enum value

`TransferAborted` enum value

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Http Member List

This is the complete list of members for `sf::Http`, including all inherited me

`Http()``sf::Http``Http(const std::string &host, unsigned short port=0)``sf::Http``NonCopyable()``sf::Nor``sendRequest(const Request &request, Time timeout=Time::Zero)` `sf::Http``setHost(const std::string &host, unsigned short port=0)``sf::Http`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Http::Request Member List

This is the complete list of members for [sf::Http::Request](#), including all inh

[Delete](#) enum value

[Get](#) enum value

[Head](#) enum value

[Http](#) (defined in [sf::Http::Request](#))

[Method](#) enum name

[Post](#) enum value

[Put](#) enum value

[Request](#)(const std::string &uri="/", Method method=Get, const std::string &body)

[setBody](#)(const std::string &body)

[setField](#)(const std::string &field, const std::string &value)

[setHttpVersion](#)(unsigned int major, unsigned int minor)

[setMethod](#)(Method method)

[setUri](#)(const std::string &uri)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Http::Response Member List

This is the complete list of members for `sf::Http::Response`, including all ir

`Accepted` enum value [sf::Http::Response](#)

`BadGateway` enum value [sf::Http::Response](#)

`BadRequest` enum value [sf::Http::Response](#)

`ConnectionFailed` enum value [sf::Http::Response](#)

`Created` enum value [sf::Http::Response](#)

`Forbidden` enum value [sf::Http::Response](#)

`GatewayTimeout` enum value [sf::Http::Response](#)

`getBody()` const [sf::Http::Response](#)

`getField(const std::string &field)` const [sf::Http::Response](#)

`getMajorHttpVersion()` const [sf::Http::Response](#)

`getMinorHttpVersion()` const [sf::Http::Response](#)

`getStatus()` const [sf::Http::Response](#)

Http (defined in `sf::Http::Response`) [sf::Http::Response](#) friend

`InternalServerError` enum value [sf::Http::Response](#)

InvalidResponse enum value	sf::Http::Response
MovedPermanently enum value	sf::Http::Response
MovedTemporarily enum value	sf::Http::Response
MultipleChoices enum value	sf::Http::Response
NoContent enum value	sf::Http::Response
NotFound enum value	sf::Http::Response
NotImplemented enum value	sf::Http::Response
NotModified enum value	sf::Http::Response
Ok enum value	sf::Http::Response
PartialContent enum value	sf::Http::Response
RangeNotSatisfiable enum value	sf::Http::Response
ResetContent enum value	sf::Http::Response
Response()	sf::Http::Response
ServiceUnavailable enum value	sf::Http::Response
Status enum name	sf::Http::Response
Unauthorized enum value	sf::Http::Response
VersionNotSupported enum value	sf::Http::Response

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::IpAddress Member List

This is the complete list of members for `sf::IpAddress`, including all inherited members.

Any	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
Broadcast	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
<code>getLocalAddress()</code>	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
<code>getPublicAddress(Time timeout=Time::Zero)</code>	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
<code>IpAddress()</code>	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
<code>IpAddress(const std::string &address)</code>	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
<code>IpAddress(const char *address)</code>	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
<code>IpAddress(Uint8 byte0, Uint8 byte1, Uint8 byte2, Uint8 byte3)</code>	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
<code>IpAddress(Uint32 address)</code>	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
LocalHost	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
None	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
<code>operator<(const IpAddress &left, const IpAddress &right)</code>	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
<code>toInteger() const</code>	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>
<code>toString() const</code>	<code>sf::IpAddress::operator<(const sf::IpAddress &left, const sf::IpAddress &right)</code>

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Packet Member List

This is the complete list of members for [sf::Packet](#), including all inherited members.

append(const void *data, std::size_t sizeInBytes)	sf::Packet
clear()	sf::Packet
endOfPacket() const	sf::Packet
getData() const	sf::Packet
getDataSize() const	sf::Packet
onReceive(const void *data, std::size_t size)	sf::Packet <small>private</small>
onSend(std::size_t &size)	sf::Packet <small>private</small>
operator BoolType() const	sf::Packet
operator<<(bool data)	sf::Packet
operator<<(Int8 data) (defined in sf::Packet)	sf::Packet
operator<<(Uint8 data) (defined in sf::Packet)	sf::Packet
operator<<(Int16 data) (defined in sf::Packet)	sf::Packet
operator<<(Uint16 data) (defined in sf::Packet)	sf::Packet
operator<<(Int32 data) (defined in sf::Packet)	sf::Packet

operator<<(Uint32 data) (defined in sf::Packet)	sf::Packet
operator<<(Int64 data) (defined in sf::Packet)	sf::Packet
operator<<(Uint64 data) (defined in sf::Packet)	sf::Packet
operator<<(float data) (defined in sf::Packet)	sf::Packet
operator<<(double data) (defined in sf::Packet)	sf::Packet
operator<<(const char *data) (defined in sf::Packet)	sf::Packet
operator<<(const std::string &data) (defined in sf::Packet)	sf::Packet
operator<<(const wchar_t *data) (defined in sf::Packet)	sf::Packet
operator<<(const std::wstring &data) (defined in sf::Packet)	sf::Packet
operator<<(const String &data) (defined in sf::Packet)	sf::Packet
operator>>(bool &data)	sf::Packet
operator>>(Int8 &data) (defined in sf::Packet)	sf::Packet
operator>>(Uint8 &data) (defined in sf::Packet)	sf::Packet
operator>>(Int16 &data) (defined in sf::Packet)	sf::Packet
operator>>(Uint16 &data) (defined in sf::Packet)	sf::Packet
operator>>(Int32 &data) (defined in sf::Packet)	sf::Packet
operator>>(Uint32 &data) (defined in sf::Packet)	sf::Packet
operator>>(Int64 &data) (defined in sf::Packet)	sf::Packet
operator>>(Uint64 &data) (defined in sf::Packet)	sf::Packet
operator>>(float &data) (defined in sf::Packet)	sf::Packet
operator>>(double &data) (defined in sf::Packet)	sf::Packet
operator>>(char *data) (defined in sf::Packet)	sf::Packet
operator>>(std::string &data) (defined in sf::Packet)	sf::Packet
operator>>(wchar_t *data) (defined in sf::Packet)	sf::Packet
operator>>(std::wstring &data) (defined in sf::Packet)	sf::Packet
operator>>(String &data) (defined in sf::Packet)	sf::Packet

Packet()	sf::Packet
TcpSocket (defined in sf::Packet)	sf::Packet fri
UdpSocket (defined in sf::Packet)	sf::Packet fri
~Packet()	sf::Packet vi

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Socket Member List

This is the complete list of members for [sf::Socket](#), including all inherited members.

AnyPort enum value	sf::Socket	
close()	sf::Socket	protected
create()	sf::Socket	protected
create(SocketHandle handle)	sf::Socket	protected
Disconnected enum value	sf::Socket	
Done enum value	sf::Socket	
Error enum value	sf::Socket	
getHandle() const	sf::Socket	protected
isBlocking() const	sf::Socket	
NonCopyable()	sf::NonCopyable	inline private
NotReady enum value	sf::Socket	
Partial enum value	sf::Socket	
setBlocking(bool blocking)	sf::Socket	
Socket(Type type)	sf::Socket	protected

SocketSelector (defined in sf::Socket)	sf::Socket	friend
Status enum name	sf::Socket	
Tcp enum value	sf::Socket	protected
Type enum name	sf::Socket	protected
Udp enum value	sf::Socket	protected
<code>-Socket()</code>	sf::Socket	virtual

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::SocketSelector Member List

This is the complete list of members for [sf::SocketSelector](#), including all in

add(Socket &socket)	sf::SocketSelector
clear()	sf::SocketSelector
isReady(Socket &socket) const	sf::SocketSelector
operator=(const SocketSelector &right)	sf::SocketSelector
remove(Socket &socket)	sf::SocketSelector
SocketSelector()	sf::SocketSelector
SocketSelector(const SocketSelector &copy)	sf::SocketSelector
wait(Time timeout=Time::Zero)	sf::SocketSelector
~SocketSelector()	sf::SocketSelector

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::TcpListener Member List

This is the complete list of members for [sf::TcpListener](#), including all inher

accept(TcpSocket &socket)	sf
AnyPort enum value	sf
close()	sf
create()	sf
create(SocketHandle handle)	sf
Disconnected enum value	sf
Done enum value	sf
Error enum value	sf
getHandle() const	sf
getLocalPort() const	sf
isBlocking() const	sf
listen(unsigned short port, const IpAddress &address=IpAddress::Any)	sf
NotReady enum value	sf
Partial enum value	sf

setBlocking(bool blocking)	sf
Socket(Type type)	sf
Status enum name	sf
Tcp enum value	sf
TcpListener()	sf
Type enum name	sf
Udp enum value	sf
~Socket()	sf

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::TcpSocket Member List

This is the complete list of members for [sf::TcpSocket](#), including all inherit

AnyPort enum value

[close\(\)](#)

[connect\(const IpAddress &remoteAddress, unsigned short remotePort, Ti](#)

[create\(\)](#)

[create\(SocketHandle handle\)](#)

[disconnect\(\)](#)

Disconnected enum value

Done enum value

Error enum value

[getHandle\(\) const](#)

[getLocalPort\(\) const](#)

[getRemoteAddress\(\) const](#)

[getRemotePort\(\) const](#)

[isBlocking\(\) const](#)

`NotReady` enum value

Partial enum value

`receive(void *data, std::size_t size, std::size_t &received)`

`receive(Packet &packet)`

`send(const void *data, std::size_t size)`

`send(const void *data, std::size_t size, std::size_t &sent)`

`send(Packet &packet)`

`setBlocking(bool blocking)`

`Socket(Type type)`

Status enum name

`Tcp` enum value

TcpListener (defined in `sf::TcpSocket`)

`TcpSocket()`

Type enum name

`Udp` enum value

`~Socket()`

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::UdpSocket Member List

This is the complete list of members for [sf::UdpSocket](#), including all inheri

AnyPort enum value

[bind\(unsigned short port, const IpAddress &address=IpAddress::Any\)](#)

[close\(\)](#)

[create\(\)](#)

[create\(SocketHandle handle\)](#)

Disconnected enum value

Done enum value

Error enum value

[getHandle\(\) const](#)

[getLocalPort\(\) const](#)

[isBlocking\(\) const](#)

MaxDatagramSize enum value

NotReady enum value

Partial enum value

```
receive(void *data, std::size_t size, std::size_t &received, IpAddress &remoteAddress)
receive(Packet &packet, IpAddress &remoteAddress, unsigned short &received)
send(const void *data, std::size_t size, const IpAddress &remoteAddress, unsigned short port)
send(Packet &packet, const IpAddress &remoteAddress, unsigned short port)
setBlocking(bool blocking)

Socket(Type type)

Status enum name

Tcp enum value

Type enum name

Udp enum value

UdpSocket()

unbind()

~Socket()
```

SFML is licensed under the terms and conditions of the zlib/png license.
Copyright © Laurent Gomila :: Documentation generated by doxygen

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Clock Member List

This is the complete list of members for `sf::Clock`, including all inherited m

[Clock\(\)](#)

sf::Clock

[getElapsedTime\(\) const](#) sf::Clock[restart\(\)](#)

sf::Clock

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::FileInputStream Member List

This is the complete list of members for `sf::FileInputStream`, including all inherited members.

<code>FileInputStream()</code>	<code>sf::FileInputStream</code>	
<code>getSize()</code>	<code>sf::FileInputStream</code>	<code>virtual</code>
<code>NonCopyable()</code>	<code>sf::NonCopyable</code>	<code>inline</code> <code>private</code>
<code>open(const std::string &filename)</code>	<code>sf::FileInputStream</code>	
<code>read(void *data, Int64 size)</code>	<code>sf::FileInputStream</code>	<code>virtual</code>
<code>seek(Int64 position)</code>	<code>sf::FileInputStream</code>	<code>virtual</code>
<code>tell()</code>	<code>sf::FileInputStream</code>	<code>virtual</code>
<code>~FileInputStream()</code>	<code>sf::FileInputStream</code>	<code>virtual</code>
<code>~InputStream()</code>	<code>sf::InputStream</code>	<code>inline</code> <code>virtual</code>

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::InputStream Member List

This is the complete list of members for [sf::InputStream](#), including all inhe

`getSize()=0` [sf::InputStream](#) pure virtual

`read(void *data, Int64 size)=0` [sf::InputStream](#) pure virtual

`seek(Int64 position)=0` [sf::InputStream](#) pure virtual

`tell()=0` [sf::InputStream](#) pure virtual

`~InputStream()` [sf::InputStream](#) inline virtual

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Lock Member List

This is the complete list of members for `sf::Lock`, including all inherited members.

`Lock(Mutex &mutex)` `sf::Lock` explicit

`NonCopyable()` `sf::NonCopyable` inline private

`~Lock()` `sf::Lock`

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::MemoryInputStream Member List

This is the complete list of members for `sf::MemoryInputStream`, including

`getSize()``sf::MemoryInputStream`

vi

`MemoryInputStream()``sf::MemoryInputStream``open(const void *data, std::size_t sizeInBytes)``sf::MemoryInputStream``read(void *data, Int64 size)``sf::MemoryInputStream`

vi

`seek(Int64 position)``sf::MemoryInputStream`

vi

`tell()``sf::MemoryInputStream`

vi

`~InputStream()``sf::InputStream`

in

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Mutex Member List

This is the complete list of members for [sf::Mutex](#), including all inherited members.

[lock\(\)](#) sf::Mutex

[Mutex\(\)](#) sf::Mutex

[NonCopyable\(\)](#) sf::NonCopyable inline private

[unlock\(\)](#) sf::Mutex

[~Mutex\(\)](#) sf::Mutex

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::NonCopyable Member List

This is the complete list of members for [sf::NonCopyable](#), including all inh

[NonCopyable\(\)](#) [sf::NonCopyable](#)

[inline](#)

[protected](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::String Member List

This is the complete list of members for [sf::String](#), including all inherited members.

[begin\(\)](#)[begin\(\) const](#)[clear\(\)](#)[ConstIterator](#) [typedef](#)[end\(\)](#)[end\(\) const](#)[erase\(std::size_t position, std::size_t count=1\)](#)[find\(const String &str, std::size_t start=0\) const](#)[fromUtf16\(T begin, T end\)](#)[fromUtf32\(T begin, T end\)](#)[fromUtf8\(T begin, T end\)](#)[getData\(\) const](#)[getSize\(\) const](#)[insert\(std::size_t position, const String &str\)](#)

InvalidPos

isEmpty() const

Iterator typedef

operator std::string() const

operator std::wstring() const

operator!=(const String &left, const String &right)

operator+(const String &left, const String &right)

operator+= (const String &right)

operator< (defined in sf::String)

operator<(const String &left, const String &right)

operator<=(const String &left, const String &right)

operator=(const String &right)

operator== (defined in sf::String)

operator==(const String &left, const String &right)

operator>(const String &left, const String &right)

operator>=(const String &left, const String &right)

operator[](std::size_t index) const

operator[](std::size_t index)

replace(std::size_t position, std::size_t length, const String &replaceWith)

replace(const String &searchFor, const String &replaceWith)

String()

String(char ansiChar, const std::locale &locale=std::locale())

String(wchar_t wideChar)

String(Uint32 utf32Char)

String(const char *ansiString, const std::locale &locale=std::locale())

```
String(const std::string &ansiString, const std::locale &locale=std::locale())
String(const wchar_t *wideString)
String(const std::wstring &wideString)
String(const Uint32 *utf32String)
String(const std::basic_string< Uint32 > &utf32String)
String(const String &copy)
substring(std::size_t position, std::size_t length=InvalidPos) const
toAnsiString(const std::locale &locale=std::locale()) const
toUtf16() const
toUtf32() const
toUtf8() const
toWideString() const
```

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Thread Member List

This is the complete list of members for `sf::Thread`, including all inherited members.

<code>launch()</code>	<code>sf::Thread</code>	
<code>NonCopyable()</code>	<code>sf::NonCopyable</code>	<small><code>inline</code> <code>private</code></small>
<code>terminate()</code>	<code>sf::Thread</code>	
<code>Thread(F function)</code>	<code>sf::Thread</code>	
<code>Thread(F function, A argument)</code>	<code>sf::Thread</code>	
<code>Thread(void(C::*function)(), C *object)</code>	<code>sf::Thread</code>	
<code>wait()</code>	<code>sf::Thread</code>	
<code>~Thread()</code>	<code>sf::Thread</code>	

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen 1.8.13

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::ThreadLocal Member List

This is the complete list of members for [sf::ThreadLocal](#), including all inhe

[getValue\(\) const](#)[sf::ThreadLocal](#)[NonCopyable\(\)](#)[sf::NonCopyable](#)[\[inline\]](#) [\[private\]](#)[setValue\(void *value\)](#)[sf::ThreadLocal](#)[ThreadLocal\(void *value=NULL\)](#) [sf::ThreadLocal](#)[~ThreadLocal\(\)](#)[sf::ThreadLocal](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::ThreadLocalPtr< T > Member List

This is the complete list of members for `sf::ThreadLocalPtr< T >`, including all inherited members.

<code>getValue() const</code>	<code>sf::ThreadLocal</code>	private
<code>operator T *() const</code>	<code>sf::ThreadLocalPtr< T ></code>	
<code>operator*() const</code>	<code>sf::ThreadLocalPtr< T ></code>	
<code>operator->() const</code>	<code>sf::ThreadLocalPtr< T ></code>	
<code>operator=(T *value)</code>	<code>sf::ThreadLocalPtr< T ></code>	
<code>operator=(const ThreadLocalPtr< T > &right)</code>	<code>sf::ThreadLocalPtr< T ></code>	
<code>setValue(void *value)</code>	<code>sf::ThreadLocal</code>	private
<code>ThreadLocal(void *value=NULL)</code>	<code>sf::ThreadLocal</code>	private
<code>ThreadLocalPtr(T *value=NULL)</code>	<code>sf::ThreadLocalPtr< T ></code>	
<code>~ThreadLocal()</code>	<code>sf::ThreadLocal</code>	private

SFML 2.4.2

Main Page | Related Pages | Modules | Namespaces | **Classes** | Files |

Class List | Class Index | Class Hierarchy | Class Members |

sf::Time Member List

This is the complete list of members for `sf::Time`, including all inherited members.

<code>asMicroseconds() const</code>	<code>sf::Time</code>	
<code>asMilliseconds() const</code>	<code>sf::Time</code>	
<code>asSeconds() const</code>	<code>sf::Time</code>	
microseconds (defined in <code>sf::Time</code>)	<code>sf::Time</code>	friend
<code>microseconds(Int64 amount)</code>	<code>sf::Time</code>	related
milliseconds (defined in <code>sf::Time</code>)	<code>sf::Time</code>	friend
<code>milliseconds(Int32 amount)</code>	<code>sf::Time</code>	related
<code>operator!=(Time left, Time right)</code>	<code>sf::Time</code>	related
<code>operator%(Time left, Time right)</code>	<code>sf::Time</code>	related
<code>operator%=(Time &left, Time right)</code>	<code>sf::Time</code>	related
<code>operator*(Time left, float right)</code>	<code>sf::Time</code>	related
<code>operator*(Time left, Int64 right)</code>	<code>sf::Time</code>	related
<code>operator*(float left, Time right)</code>	<code>sf::Time</code>	related
<code>operator*(Int64 left, Time right)</code>	<code>sf::Time</code>	related

<code>operator*=(Time &left, float right)</code>	<code>sf::Time</code>	related
<code>operator*=(Time &left, Int64 right)</code>	<code>sf::Time</code>	related
<code>operator+(Time left, Time right)</code>	<code>sf::Time</code>	related
<code>operator+=(Time &left, Time right)</code>	<code>sf::Time</code>	related
<code>operator-(Time right)</code>	<code>sf::Time</code>	related
<code>operator-(Time left, Time right)</code>	<code>sf::Time</code>	related
<code>operator-=(Time &left, Time right)</code>	<code>sf::Time</code>	related
<code>operator/(Time left, float right)</code>	<code>sf::Time</code>	related
<code>operator/(Time left, Int64 right)</code>	<code>sf::Time</code>	related
<code>operator/(Time left, Time right)</code>	<code>sf::Time</code>	related
<code>operator/=(Time &left, float right)</code>	<code>sf::Time</code>	related
<code>operator/=(Time &left, Int64 right)</code>	<code>sf::Time</code>	related
<code>operator<(Time left, Time right)</code>	<code>sf::Time</code>	related
<code>operator<=(Time left, Time right)</code>	<code>sf::Time</code>	related
<code>operator==(Time left, Time right)</code>	<code>sf::Time</code>	related
<code>operator>(Time left, Time right)</code>	<code>sf::Time</code>	related
<code>operator>=(Time left, Time right)</code>	<code>sf::Time</code>	related
seconds (defined in <code>sf::Time</code>)	<code>sf::Time</code>	friend
<code>seconds(float amount)</code>	<code>sf::Time</code>	related
<code>Time()</code>	<code>sf::Time</code>	
<code>Zero</code>	<code>sf::Time</code>	static

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Vector2< T > Member List

This is the complete list of members for `sf::Vector2< T >`, including all inheritance-based members.

<code>operator!=</code> (const Vector2< T > &left, const Vector2< T > &right)	<code>sf::Vector2< T ></code>
<code>operator*</code> (const Vector2< T > &left, T right)	<code>sf::Vector2< T ></code>
<code>operator*(T left, const Vector2< T > &right)</code>	<code>sf::Vector2< T ></code>
<code>operator*=(Vector2< T > &left, T right)</code>	<code>sf::Vector2< T ></code>
<code>operator+(const Vector2< T > &left, const Vector2< T > &right)</code>	<code>sf::Vector2< T ></code>
<code>operator+=(Vector2< T > &left, const Vector2< T > &right)</code>	<code>sf::Vector2< T ></code>
<code>operator-(const Vector2< T > &right)</code>	<code>sf::Vector2< T ></code>
<code>operator-(const Vector2< T > &left, const Vector2< T > &right)</code>	<code>sf::Vector2< T ></code>
<code>operator-=(Vector2< T > &left, const Vector2< T > &right)</code>	<code>sf::Vector2< T ></code>
<code>operator/(const Vector2< T > &left, T right)</code>	<code>sf::Vector2< T ></code>
<code>operator/=(Vector2< T > &left, T right)</code>	<code>sf::Vector2< T ></code>
<code>operator==(const Vector2< T > &left, const Vector2< T > &right)</code>	<code>sf::Vector2< T ></code>
<code>Vector2()</code>	<code>sf::Vector2< T ></code>
<code>Vector2(T X, T Y)</code>	<code>sf::Vector2< T ></code>

Vector2(const Vector2< U > &vector)	sf::Vector2< U >
x	sf::Vector2< U >
y	sf::Vector2< U >

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Vector3< T > Member List

This is the complete list of members for `sf::Vector3< T >`, including all inheritance-based members.

<code>operator!=</code> (const Vector3< T > &left, const Vector3< T > &right)	<code>sf::Vector3< T ></code>
<code>operator*</code> (const Vector3< T > &left, T right)	<code>sf::Vector3< T ></code>
<code>operator*(T left, const Vector3< T > &right)</code>	<code>sf::Vector3< T ></code>
<code>operator*=(Vector3< T > &left, T right)</code>	<code>sf::Vector3< T ></code>
<code>operator+(const Vector3< T > &left, const Vector3< T > &right)</code>	<code>sf::Vector3< T ></code>
<code>operator+=(Vector3< T > &left, const Vector3< T > &right)</code>	<code>sf::Vector3< T ></code>
<code>operator-(const Vector3< T > &left)</code>	<code>sf::Vector3< T ></code>
<code>operator-(const Vector3< T > &left, const Vector3< T > &right)</code>	<code>sf::Vector3< T ></code>
<code>operator-=(Vector3< T > &left, const Vector3< T > &right)</code>	<code>sf::Vector3< T ></code>
<code>operator/(const Vector3< T > &left, T right)</code>	<code>sf::Vector3< T ></code>
<code>operator/=(Vector3< T > &left, T right)</code>	<code>sf::Vector3< T ></code>
<code>operator==(const Vector3< T > &left, const Vector3< T > &right)</code>	<code>sf::Vector3< T ></code>
<code>Vector3()</code>	<code>sf::Vector3< T ></code>
<code>Vector3(T X, T Y, T Z)</code>	<code>sf::Vector3< T ></code>

Vector3(const Vector3< U > &vector)	sf::Vector3< U >
x	sf::Vector3< U >
y	sf::Vector3< U >
z	sf::Vector3< U >

SFML is licensed under the terms and conditions of the zlib/png license.
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Context Member List

This is the complete list of members for [sf::Context](#), including all inherited

[Context\(\)](#)[Context\(const ContextSettings &settings, unsigned int width, unsigned int ensureGIContext\(\)\)](#)[getActiveContext\(\)](#)[getFunction\(const char *name\)](#)[getSettings\(\) const](#)[GIResource\(\)](#)[isExtensionAvailable\(const char *name\)](#)[NonCopyable\(\)](#)[setActive\(bool active\)](#)[~Context\(\)](#)[~GIResource\(\)](#)

Copyright © Laurent Gomila :: Documentation generated by d

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::ContextSettings Member List

This is the complete list of members for [sf::ContextSettings](#), including all inherited members.

[antialiasingLevel](#)

Attribute enum name

[attributeFlags](#)

ContextSettings(unsigned int depth=0, unsigned int stencil=0, unsigned int antialiasingLevel=0, unsigned int attributeFlags=0, sf::Core core=Core::Default, sf::Debug debug=Debug::Default, sf::MajorVersion majorVersion=MajorVersion::Default, sf::MinorVersion minorVersion=MinorVersion::Default, bool sRgbCapable=false)

Core enum value

Debug enum value

Default enum value

depthBits

majorVersion

minorVersion

sRgbCapable

stencilBits

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Event Member List

This is the complete list of members for `sf::Event`, including all inherited members.

Closed	enum value	<code>sf::Event</code>
Count	enum value	<code>sf::Event</code>
EventType	enum name	<code>sf::Event</code>
GainedFocus	enum value	<code>sf::Event</code>
joystickButton		<code>sf::Event</code>
JoystickButtonPressed	enum value	<code>sf::Event</code>
JoystickButtonReleased	enum value	<code>sf::Event</code>
joystickConnect		<code>sf::Event</code>
JoystickConnected	enum value	<code>sf::Event</code>
JoystickDisconnected	enum value	<code>sf::Event</code>
joystickMove		<code>sf::Event</code>
JoystickMoved	enum value	<code>sf::Event</code>
key		<code>sf::Event</code>
KeyPressed	enum value	<code>sf::Event</code>

KeyReleased enum value	sf::Event
LostFocus enum value	sf::Event
mouseButton	sf::Event
MouseButtonPressed enum value	sf::Event
MouseButtonReleased enum value	sf::Event
MouseEntered enum value	sf::Event
MouseLeft enum value	sf::Event
mouseMove	sf::Event
MouseMoved enum value	sf::Event
mouseWheel	sf::Event
MouseWheelMoved enum value	sf::Event
mouseWheelScroll	sf::Event
MouseWheelScrolled enum value	sf::Event
Resized enum value	sf::Event
sensor	sf::Event
SensorChanged enum value	sf::Event
size	sf::Event
text	sf::Event
TextEntered enum value	sf::Event
touch	sf::Event
TouchBegan enum value	sf::Event
TouchEnded enum value	sf::Event
TouchMoved enum value	sf::Event
type	sf::Event

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::JoystickButtonEvent Member List

This is the complete list of members for [sf::Event::JoystickButtonEvent](#), in

[button](#) [sf::Event::JoystickButtonEvent](#)

[joystickId](#) [sf::Event::JoystickButtonEvent](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::JoystickConnectEvent Member List

This is the complete list of members for [sf::Event::JoystickConnectEvent](#),

[joystickId](#) [sf::Event::JoystickConnectEvent](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::JoystickMoveEvent Member List

This is the complete list of members for `sf::Event::JoystickMoveEvent`, including all inherited members.

axis `sf::Event::JoystickMoveEvent`

joystickId `sf::Event::JoystickMoveEvent`

position `sf::Event::JoystickMoveEvent`

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::KeyEvent Member List

This is the complete list of members for [sf::Event::KeyEvent](#), including all

alt [sf::Event::KeyEvent](#)

code [sf::Event::KeyEvent](#)

control [sf::Event::KeyEvent](#)

shift [sf::Event::KeyEvent](#)

system [sf::Event::KeyEvent](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::MouseEvent Member List

This is the complete list of members for [sf::Event::MouseEvent](#), including all inherited members.

button [sf::Event::MouseEvent](#)

x [sf::Event::MouseEvent](#)

y [sf::Event::MouseEvent](#)

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::MouseMoveEvent Member List

This is the complete list of members for [sf::Event::MouseMoveEvent](#), including all inherited members.

x [sf::Event::MouseMoveEvent](#)

y [sf::Event::MouseMoveEvent](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::MouseWheelEvent Member List

This is the complete list of members for [sf::Event::MouseWheelEvent](#), including all inherited members.

delta [sf::Event::MouseWheelEvent](#)

x [sf::Event::MouseWheelEvent](#)

y [sf::Event::MouseWheelEvent](#)

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::MouseWheelScrollEvent Member

This is the complete list of members for `sf::Event::MouseWheelScrollEvent` members.

<code>delta</code>	<code>sf::Event::MouseWheelScrollEvent</code>
<code>wheel</code>	<code>sf::Event::MouseWheelScrollEvent</code>
<code>x</code>	<code>sf::Event::MouseWheelScrollEvent</code>
<code>y</code>	<code>sf::Event::MouseWheelScrollEvent</code>

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::SensorEvent Member List

This is the complete list of members for [sf::Event::SensorEvent](#), including

[type sf::Event::SensorEvent](#)

[x sf::Event::SensorEvent](#)

[y sf::Event::SensorEvent](#)

[z sf::Event::SensorEvent](#)

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::SizeEvent Member List

This is the complete list of members for [sf::Event::SizeEvent](#), including all

[height sf::Event::SizeEvent](#)

[width sf::Event::SizeEvent](#)

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::TextEvent Member List

This is the complete list of members for [sf::Event::TextEvent](#), including all

[unicode](#) [sf::Event::TextEvent](#)

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Event::TouchEvent Member List

This is the complete list of members for `sf::Event::TouchEvent`, including all inherited members.

`finger` `sf::Event::TouchEvent`

`x` `sf::Event::TouchEvent`

`y` `sf::Event::TouchEvent`

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::GIResource Member List

This is the complete list of members for [sf::GIResource](#), including all inherited members.

[ensureGIContext\(\)](#) sf::GIResource protected static

[GIResource\(\)](#) sf::GIResource protected

[~GIResource\(\)](#) sf::GIResource protected

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

`sf::GIResource::TransientContextLock` Member List

This is the complete list of members for `sf::GIResource::TransientContextLock`. The list is organized by member type and is indexed by name.

[NonCopyable\(\)](#)`sf::NonCopyable`

inline

private

[TransientContextLock\(\)](#)`sf::GIResource::TransientContextLock`[~TransientContextLock\(\)](#)`sf::GIResource::TransientContextLock`

sf::Joystick Member List

This is the complete list of members for [sf::Joystick](#), including all inherited

Axis enum name	sf::Joystick
AxisCount enum value	sf::Joystick
ButtonCount enum value	sf::Joystick
Count enum value	sf::Joystick
getAxisPosition(unsigned int joystick, Axis axis)	sf::Joystick st
getButtonCount(unsigned int joystick)	sf::Joystick st
getIdentification(unsigned int joystick)	sf::Joystick st
hasAxis(unsigned int joystick, Axis axis)	sf::Joystick st
isButtonPressed(unsigned int joystick, unsigned int button)	sf::Joystick st
isConnected(unsigned int joystick)	sf::Joystick st
PovX enum value	sf::Joystick
PovY enum value	sf::Joystick
R enum value	sf::Joystick
U enum value	sf::Joystick

update()	sf::Joystick
V enum value	sf::Joystick
X enum value	sf::Joystick
Y enum value	sf::Joystick
Z enum value	sf::Joystick

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Joystick::Identification Member List

This is the complete list of members for [sf::Joystick::Identification](#), includir

Identification() (defined in sf::Joystick::Identification)	sf::Joystick::Identifi
name	sf::Joystick::Identifi
productId	sf::Joystick::Identifi
vendorId	sf::Joystick::Identifi

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Keyboard Member List

This is the complete list of members for `sf::Keyboard`, including all inheritance.

A enum value [sf::Keyboard](#)

Add enum value [sf::Keyboard](#)

B enum value [sf::Keyboard](#)

BackSlash enum value [sf::Keyboard](#)

BackSpace enum value [sf::Keyboard](#)

C enum value [sf::Keyboard](#)

Comma enum value [sf::Keyboard](#)

D enum value [sf::Keyboard](#)

Dash enum value [sf::Keyboard](#)

Delete enum value [sf::Keyboard](#)

Divide enum value [sf::Keyboard](#)

Down enum value [sf::Keyboard](#)

E enum value [sf::Keyboard](#)

End enum value [sf::Keyboard](#)

Equal enum value	sf::Keyboard
Escape enum value	sf::Keyboard
F enum value	sf::Keyboard
F1 enum value	sf::Keyboard
F10 enum value	sf::Keyboard
F11 enum value	sf::Keyboard
F12 enum value	sf::Keyboard
F13 enum value	sf::Keyboard
F14 enum value	sf::Keyboard
F15 enum value	sf::Keyboard
F2 enum value	sf::Keyboard
F3 enum value	sf::Keyboard
F4 enum value	sf::Keyboard
F5 enum value	sf::Keyboard
F6 enum value	sf::Keyboard
F7 enum value	sf::Keyboard
F8 enum value	sf::Keyboard
F9 enum value	sf::Keyboard
G enum value	sf::Keyboard
H enum value	sf::Keyboard
Home enum value	sf::Keyboard
I enum value	sf::Keyboard
Insert enum value	sf::Keyboard
isKeyPressed(Key key)	sf::Keyboard
J enum value	sf::Keyboard
K enum value	sf::Keyboard

Key enum name	sf::Keyboard
KeyCount enum value	sf::Keyboard
L enum value	sf::Keyboard
LAlt enum value	sf::Keyboard
LBracket enum value	sf::Keyboard
LControl enum value	sf::Keyboard
Left enum value	sf::Keyboard
LShift enum value	sf::Keyboard
LSystem enum value	sf::Keyboard
M enum value	sf::Keyboard
Menu enum value	sf::Keyboard
Multiply enum value	sf::Keyboard
N enum value	sf::Keyboard
Num0 enum value	sf::Keyboard
Num1 enum value	sf::Keyboard
Num2 enum value	sf::Keyboard
Num3 enum value	sf::Keyboard
Num4 enum value	sf::Keyboard
Num5 enum value	sf::Keyboard
Num6 enum value	sf::Keyboard
Num7 enum value	sf::Keyboard
Num8 enum value	sf::Keyboard
Num9 enum value	sf::Keyboard
Numpad0 enum value	sf::Keyboard
Numpad1 enum value	sf::Keyboard

Numpad2 enum value	sf::Keyboard
Numpad3 enum value	sf::Keyboard
Numpad4 enum value	sf::Keyboard
Numpad5 enum value	sf::Keyboard
Numpad6 enum value	sf::Keyboard
Numpad7 enum value	sf::Keyboard
Numpad8 enum value	sf::Keyboard
Numpad9 enum value	sf::Keyboard
O enum value	sf::Keyboard
P enum value	sf::Keyboard
PageDown enum value	sf::Keyboard
PageUp enum value	sf::Keyboard
Pause enum value	sf::Keyboard
Period enum value	sf::Keyboard
Q enum value	sf::Keyboard
Quote enum value	sf::Keyboard
R enum value	sf::Keyboard
RAlt enum value	sf::Keyboard
RBracket enum value	sf::Keyboard
RControl enum value	sf::Keyboard
Return enum value	sf::Keyboard
Right enum value	sf::Keyboard
RShift enum value	sf::Keyboard
RSystem enum value	sf::Keyboard
S enum value	sf::Keyboard
SemiColon enum value	sf::Keyboard

<code>setVirtualKeyboardVisible(bool visible)</code>	<code>sf::Keyboard</code>	<code>static</code>
<code>Slash</code> enum value	<code>sf::Keyboard</code>	
<code>Space</code> enum value	<code>sf::Keyboard</code>	
<code>Subtract</code> enum value	<code>sf::Keyboard</code>	
<code>T</code> enum value	<code>sf::Keyboard</code>	
<code>Tab</code> enum value	<code>sf::Keyboard</code>	
<code>Tilde</code> enum value	<code>sf::Keyboard</code>	
<code>U</code> enum value	<code>sf::Keyboard</code>	
<code>Unknown</code> enum value	<code>sf::Keyboard</code>	
<code>Up</code> enum value	<code>sf::Keyboard</code>	
<code>V</code> enum value	<code>sf::Keyboard</code>	
<code>W</code> enum value	<code>sf::Keyboard</code>	
<code>X</code> enum value	<code>sf::Keyboard</code>	
<code>Y</code> enum value	<code>sf::Keyboard</code>	
<code>Z</code> enum value	<code>sf::Keyboard</code>	

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Mouse Member List

This is the complete list of members for [sf::Mouse](#), including all inherited members.

Button	enum name	sf::Mouse::Button
ButtonCount	enum value	sf::Mouse::ButtonCount
getPosition()		sf::Mouse::getPosition()
getPosition(const Window &relativeTo)		sf::Mouse::getPosition(const Window &relativeTo)
HorizontalWheel	enum value	sf::Mouse::HorizontalWheel
isButtonPressed(Button button)		sf::Mouse::isButtonPressed(Button button)
Left	enum value	sf::Mouse::Left
Middle	enum value	sf::Mouse::Middle
Right	enum value	sf::Mouse::Right
setPosition(const Vector2i &position)		sf::Mouse::setPosition(const Vector2i &position)
setPosition(const Vector2i &position, const Window &relativeTo)		sf::Mouse::setPosition(const Vector2i &position, const Window &relativeTo)
VerticalWheel	enum value	sf::Mouse::VerticalWheel
Wheel	enum name	sf::Mouse::Wheel
XButton1	enum value	sf::Mouse::XButton1

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

sf::Sensor Member List

This is the complete list of members for [sf::Sensor](#), including all inherited members.

Accelerometer enum value	sf::Sensor
Count enum value	sf::Sensor
getValue(Type sensor)	sf::Sensor static
Gravity enum value	sf::Sensor
Gyroscope enum value	sf::Sensor
isAvailable(Type sensor)	sf::Sensor static
Magnetometer enum value	sf::Sensor
Orientation enum value	sf::Sensor
setEnabled(Type sensor, bool enabled)	sf::Sensor static
Type enum name	sf::Sensor
UserAcceleration enum value	sf::Sensor

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Touch Member List

This is the complete list of members for [sf::Touch](#), including all inherited members.

[getPosition\(unsigned int finger\)](#)sf::Touch static[getPosition\(unsigned int finger, const Window &relativeTo\)](#) sf::Touch static[isDown\(unsigned int finger\)](#)sf::Touch static

SFML is licensed under the terms and conditions of the zlib/png license.

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::VideoMode Member List

This is the complete list of members for `sf::VideoMode`, including all inheri

`bitsPerPixel`

`getDesktopMode()`

`getFullscreenModes()`

`height`

`isValid() const`

`operator!= (const VideoMode &left, const VideoMode &right)`

`operator< (const VideoMode &left, const VideoMode &right)`

`operator<= (const VideoMode &left, const VideoMode &right)`

`operator== (const VideoMode &left, const VideoMode &right)`

`operator> (const VideoMode &left, const VideoMode &right)`

`operator>= (const VideoMode &left, const VideoMode &right)`

`VideoMode()`

`VideoMode(unsigned int modeWidth, unsigned int modeHeight, unsigned width)`

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[**Classes**](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Window Member List

This is the complete list of members for [sf::Window](#), including all inherited

[close\(\)](#)[create\(VideoMode mode, const String &title, Uint32 style=Style::Default, const ContextSettings &settings=ContextSettings\(\)\)](#)[create\(WindowHandle handle, const ContextSettings &settings=ContextSettings\(\)\)](#)[display\(\)](#)[ensureGLContext\(\)](#)[getPosition\(\) const](#)[getSettings\(\) const](#)[getSize\(\) const](#)[getSystemHandle\(\) const](#)[GIResource\(\)](#)[hasFocus\(\) const](#)[isOpen\(\) const](#)[NonCopyable\(\)](#)[onCreate\(\)](#)

```
onResize()
pollEvent(Event &event)
requestFocus()
setActive(bool active=true) const
setFramerateLimit(unsigned int limit)
setIcon(unsigned int width, unsigned int height, const Uint8 *pixels)
setJoystickThreshold(float threshold)
setKeyRepeatEnabled(bool enabled)
setMouseCursorGrabbed(bool grabbed)
setMouseCursorVisible(bool visible)
setPosition(const Vector2i &position)
setSize(const Vector2u &size)
setTitle(const String &title)
setVerticalSyncEnabled(bool enabled)
setVisible(bool visible)
waitEvent(Event &event)
Window()
Window(VideoMode mode, const String &title, Uint32 style=Style::Default)
Window(WindowHandle handle, const ContextSettings &settings=ContextSettings())
~GIResource()
~Window()
```

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Utf< 16 > Member List

This is the complete list of members for `sf::Utf< 16 >`, including all inheritance.

`count(In begin, In end)`

`decode(In begin, In end, Uint32 &output, Uint32 replacement=0)`

`encode(Uint32 input, Out output, Uint16 replacement=0)`

`fromAnsi(In begin, In end, Out output, const std::locale &locale=std::locale::classic())`

`fromLatin1(In begin, In end, Out output)`

`fromWide(In begin, In end, Out output)`

`next(In begin, In end)`

`toAnsi(In begin, In end, Out output, char replacement=0, const std::locale &locale=std::locale::classic())`

`toLatin1(In begin, In end, Out output, char replacement=0)`

`toUtf16(In begin, In end, Out output)`

`toUtf32(In begin, In end, Out output)`

`toUtf8(In begin, In end, Out output)`

`toWide(In begin, In end, Out output, wchar_t replacement=0)`

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | **Classes** | [Files](#)

[Class List](#) | [Class Index](#) | [Class Hierarchy](#) | [Class Members](#)

sf::Utf< 32 > Member List

This is the complete list of members for `sf::Utf< 32 >`, including all inheritance.

`count(In begin, In end)`

`decode(In begin, In end, Uint32 &output, Uint32 replacement=0)`

`decodeAnsi(In input, const std::locale &locale=std::locale())`

`decodeWide(In input)`

`encode(Uint32 input, Out output, Uint32 replacement=0)`

`encodeAnsi(Uint32 codepoint, Out output, char replacement=0, const std::`

`encodeWide(Uint32 codepoint, Out output, wchar_t replacement=0)`

`fromAnsi(In begin, In end, Out output, const std::locale &locale=std::locale())`

`fromLatin1(In begin, In end, Out output)`

`fromWide(In begin, In end, Out output)`

`next(In begin, In end)`

`toAnsi(In begin, In end, Out output, char replacement=0, const std::locale &locale=std::locale())`

`toLatin1(In begin, In end, Out output, char replacement=0)`

`toUtf16(In begin, In end, Out output)`

`toUtf32(In begin, In end, Out output)`

`toUtf8(In begin, In end, Out output)`

`toWide(In begin, In end, Out output, wchar_t replacement=0)`

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)

sf::Utf< 8 > Member List

This is the complete list of members for `sf::Utf< 8 >`, including all inherited

`count(In begin, In end)`

`decode(In begin, In end, Uint32 &output, Uint32 replacement=0)`

`encode(Uint32 input, Out output, Uint8 replacement=0)`

`fromAnsi(In begin, In end, Out output, const std::locale &locale=std::locale::classic())`

`fromLatin1(In begin, In end, Out output)`

`fromWide(In begin, In end, Out output)`

`next(In begin, In end)`

`toAnsi(In begin, In end, Out output, char replacement=0, const std::locale &locale=std::locale::classic())`

`toLatin1(In begin, In end, Out output, char replacement=0)`

`toUtf16(In begin, In end, Out output)`

`toUtf32(In begin, In end, Out output)`

`toUtf8(In begin, In end, Out output)`

`toWide(In begin, In end, Out output, wchar_t replacement=0)`

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

include Directory Reference

Directories

directory **SFML**

SFML is licensed under the terms and conditions of the [zlib/pn](#)

Copyright © Laurent Gomila :: Documentation generated by [doxygen](#)

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | [Classes](#) | [Files](#)

SFML Directory Reference

Directories

directory `Audio`

directory `Graphics`

directory `Network`

directory `System`

directory `Window`

Files

file **Audio.hpp** [code]

file **Config.hpp** [code]

file **Graphics.hpp** [code]

file **Main.hpp** [code]

file **Network.hpp** [code]

file **OpenGL.hpp** [code]

file **System.hpp** [code]

file **Window.hpp** [code]

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

Audio Directory Reference

Files

file **AIResource.hpp** [code]

file **Audio/Export.hpp** [code]

file **InputSoundFile.hpp** [code]

file **Listener.hpp** [code]

file **Music.hpp** [code]

file **OutputSoundFile.hpp** [code]

file **Sound.hpp** [code]

file **SoundBuffer.hpp** [code]

file **SoundBufferRecorder.hpp** [code]

file **SoundFileFactory.hpp** [code]

file **SoundFileReader.hpp** [code]

file **SoundFileWriter.hpp** [code]

file **SoundRecorder.hpp** [code]

file **SoundSource.hpp** [code]

file **SoundStream.hpp** [code]

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#) | [Related Pages](#) | [Modules](#) | [Namespaces](#) | [Classes](#) | [Files](#)

Graphics Directory Reference

Files

file **BlendMode.hpp** [code]

file **CircleShape.hpp** [code]

file **Color.hpp** [code]

file **ConvexShape.hpp** [code]

file **Drawable.hpp** [code]

file **Graphics/Export.hpp** [code]

file **Font.hpp** [code]

file **GIsI.hpp** [code]

file **Glyph.hpp** [code]

file **Image.hpp** [code]

file **PrimitiveType.hpp** [code]

file **Rect.hpp** [code]

file **RectangleShape.hpp** [code]

file **RenderStates.hpp** [code]

file **RenderTarget.hpp** [code]

file **RenderTexture.hpp** [code]

file **RenderWindow.hpp** [code]

file **Shader.hpp** [code]

file **Shape.hpp** [code]

file **Sprite.hpp** [code]

file **Text.hpp** [code]

file **Texture.hpp** [code]

file **Transform.hpp** [code]

file **Transformable.hpp** [code]

file **Vertex.hpp** [code]

file **VertexArray.hpp** [code]

file **View.hpp** [code]

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

System Directory Reference

Files

file **Clock.hpp** [code]

file **Err.hpp** [code]

file **System/Export.hpp** [code]

file **FileInputStream.hpp** [code]

file **InputStream.hpp** [code]

file **Lock.hpp** [code]

file **MemoryInputStream.hpp** [code]

file **Mutex.hpp** [code]

file **NativeActivity.hpp** [code]

file **NonCopyable.hpp** [code]

file **Sleep.hpp** [code]

file **String.hpp** [code]

file **Thread.hpp** [code]

file **ThreadLocal.hpp** [code]

file **ThreadLocalPtr.hpp** [code]

file **Time.hpp** [code]

file **Utf.hpp** [code]

file **Vector2.hpp** [code]

file **Vector3.hpp** [code]

SFML is licensed under the terms and conditions of the zlib/pn

Copyright © Laurent Gomila :: Documentation generated by d

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

Window Directory Reference

Files

file **Context.hpp** [code]

file **ContextSettings.hpp** [code]

file **Event.hpp** [code]

file **Window/Export.hpp** [code]

file **GIResource.hpp** [code]

file **Joystick.hpp** [code]

file **Keyboard.hpp** [code]

file **Mouse.hpp** [code]

file **Sensor.hpp** [code]

file **Touch.hpp** [code]

file **VideoMode.hpp** [code]

file **Window/Window.hpp** [code]

file **WindowHandle.hpp** [code]

file **WindowStyle.hpp** [code]

Copyright © Laurent Gomila :: Documentation generated by doxygen

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

Network Directory Reference

Files

file **Network/Export.hpp** [code]

file **Ftp.hpp** [code]

file **Http.hpp** [code]

file **IpAddress.hpp** [code]

file **Packet.hpp** [code]

file **Socket.hpp** [code]

file **SocketHandle.hpp** [code]

file **SocketSelector.hpp** [code]

file **TcpListener.hpp** [code]

file **TcpSocket.hpp** [code]

file **UdpSocket.hpp** [code]

SFML 2.4.2

[Main Page](#)

[Related Pages](#)

[Modules](#)

[Namespaces](#)

[Classes](#)

[Files](#)

doc Directory Reference

Files

file **mainpage.hpp** [code]

SFML is licensed under the terms and conditions of the zlib/pn
Copyright © Laurent Gomila :: Documentation generated by d